PEMOGRAMAN BERORIENTASI OBJEK

Semester

1

Untuk SMK / MAK Kelas XI

Penulis : **Eko Subiyantoro**
Editor Materi : **Joko Pitono**
Editor Bahasa :
Ilustrasi Sampul :
Desain & Ilustrasi Buku : **PPPPTK BOE Malang**

Pusat Pengembangan & Pemberdayaan Pendidik & Tenaga Kependidikan Bidang Otomotif & Elektronika:

Jl. Teluk Mandar, Arjosari Tromol Pos 5, Malang 65102, Telp. (0341) 491239, (0341) 495849, Fax. (0341) 491342, Surel: vedcmalang@vedcmalang.or.id, Laman: www.vedcmalang.com

DISKLAIMER (*DISCLAIMER*)

Penerbit tidak menjamin kebenaran dan keakuratan isi/informasi yang tertulis di dalam buku tek ini. Kebenaran dan keakuratan isi/informasi merupakan tanggung jawab dan wewenang dari penulis.

Penerbit tidak bertanggung jawab dan tidak melayani terhadap semua komentar apapun yang ada didalam buku teks ini. Setiap komentar yang tercantum untuk tujuan perbaikan isi adalah tanggung jawab dari masing-masing penulis.

Setiap kutipan yang ada di dalam buku teks akan dicantumkan sumbernya dan penerbit tidak bertanggung jawab terhadap isi dari kutipan tersebut. Kebenaran keakuratan isi kutipan tetap menjadi tanggung jawab dan hak diberikan pada penulis dan pemilik asli. Penulis bertanggung jawab penuh terhadap setiap perawatan (perbaikan) dalam menyusun informasi dan bahan dalam buku teks ini.

Penerbit tidak bertanggung jawab atas kerugian, kerusakan atau ketidaknyamanan yang disebabkan sebagai akibat dari ketidakjelasan, ketidaktepatan atau kesalahan didalam menyusun makna kalimat didalam buku teks ini.

Kewenangan Penerbit hanya sebatas memindahkan atau menerbitkan mempublikasi, mencetak, memegang dan memproses data sesuai dengan undang-undang yang berkaitan dengan perlindungan data.

Katalog Dalam Terbitan (KDT)
Teknik Komunikasi dan Informatika Edisi Pertama 2013
Kementerian Pendidikan & Kebudayaan
Direktorat Jenderal Peningkatan Mutu Pendidik & Tenaga Kependidikan,
th. 2013: Jakarta

KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan yang Maha Esa atas tersusunnya buku teks ini, dengan harapan dapat digunakan sebagai buku teks untuk siswa Sekolah Menengah Kejuruan (SMK) Bidang Studi Keahlian Teknologi Informasi dan Komunikasi, Program Keahlian Teknik Komputer dan Informatika.

Penerapan kurikulum 2013 mengacu pada paradigma belajar kurikulum abad 21 menyebabkan terjadinya perubahan, yakni dari pengajaran (*teaching*) menjadi BELAJAR (*learning*), dari pembelajaran yang berpusat kepada guru (*teachers-centered*) menjadi pembelajaran yang berpusat kepada peserta didik (*student-centered*), dari pembelajaran pasif (*pasive learning*) ke cara belajar peserta didik aktif (*active learning-CBSA*) atau *Student Active Learning-SAL*.

Buku teks Pemrograman Berorientasi Obyek ini disusun berdasarkan tuntutan paradigma pengajaran dan pembelajaran kurikulum 2013 diselaraskan berdasarkan pendekatan model pembelajaran yang sesuai dengan kebutuhan belajar kurikulum abad 21, yaitu pendekatan model pembelajaran berbasis peningkatan keterampilan proses sains.

Penyajian buku teks untuk Mata Pelajaran Pemrograman Berorientasi Obyek ini disusun dengan tujuan agar supaya peserta didik dapat melakukan proses pencarian pengetahuan berkenaan dengan materi pelajaran melalui berbagai aktivitas proses sains sebagaimana dilakukan oleh para ilmuwan dalam melakukan eksperimen ilmiah (penerapan scientifik), dengan demikian peserta didik diarahkan untuk menemukan sendiri berbagai fakta, membangun konsep, dan nilai-nilai baru secara mandiri.

Kementerian Pendidikan dan Kebudayaan, Direktorat Pembinaan Sekolah Menengah Kejuruan, dan Direktorat Jenderal Peningkatan Mutu Pendidik dan Tenaga Kependidikan menyampaikan terima kasih, sekaligus saran kritik demi kesempurnaan buku teks ini dan penghargaan kepada semua pihak yang telah berperan serta dalam membantu terselesaikannya buku teks siswa untuk Mata Pelajaran Pemrograman Berorientasi Obyek kelas XI/Semester 1 Sekolah Menengah Kejuruan (SMK).

Jakarta, 12 Desember 2013
Menteri Pendidikan dan Kebudayaan

Prof. Dr. Mohammad Nuh, DEA

DAFTAR ISI

GLOSARIUM

Abstraksi mengacu pada tindakan yang mewakili fitur penting tanpa termasuk rincian latar belakang atau penjelasan. Kelas menggunakan konsep abstraksi dan didefinisikan sebagai daftar atribut abstrak.

Accessor Methods digunakan untuk membaca nilai variabel pada class,baik berupa instance maupun static. Sebuah accessor method umumnya dimulai dengan penulisan get <namaInstanceVariable>.

Array adalah sebuah variabel/ sebuah lokasi tertentu yang memiliki satu nama sebagai identifier, namun identifier ini dapat menyimpan lebih dari sebuah nilai.

Encapsulation adalah mekanisme yang mengikat bersama-sama kode dan data dalam memanipulasi, dan membuat baik aman dari gangguan luar dan penyalahgunaan. Enkapsulasi merupakan penyimpanan data dan fungsi dalam satu unit (kelas). Sebuah antarmuka yang terdefinisi dengan baik mengontrol akses ke kode tertentu dan data. Data tidak bisa diakses oleh dunia luar dan hanya fungsi-fungsi yang disimpan dalam kelas dapat mengaksesnya.

Field merupakan tipe data yang didefiniskan, sementara method merupakan operasi.

Inheritance (pewarisan) adalah proses dimana satu objek mengakuisisi properti dari obyek lain. Ini mendukung klasifikasi hirarkis. Dengan menggunakan warisan, objek hanya perlu mendefinisikan kualitas-kualitas yang membuatnya unik dalam kelasnya. Hal ini dapat mewarisi atribut umum dari induknya. Sebuah sub - class baru mewarisi semua atribut dari super - class nya.

Java SDK adalah platform dasar java yang diperlukan agar komputer atau laptop dapat digunakan untuk mengeksekusi kode-kode program bahasa java.

Kelas adalah kumpulan objek dari jenis yang sama. Setelah kelas didefinisikan, sejumlah objek dapat dibuat yang termasuk ke kelas tersebut. Kelas adalah cetak biru, atau prototipe, yang mendefinisikan variabel dan metode umum untuk semua objek dari jenis tertentu.

Metode menggambarkan kemampuan objek . Anjing memiliki kemampuan untuk kulit. Jadi kulit ( ) adalah salah satu metode dari kelas Dog.

Mutator methods adalah *method* yang dapat memberi atau mengubah nilai variable dalam *class*,baik itu berupa *instance* maupun *static*.

Netbeans adalah aplikasi editor terpadu (IDE atau Integreted Development Environment) yang akan mempermudah dalam membuat aplikasi karena menyediakan control visual yang penting dalam pemrograman desktop (pemrograman visual).

Objek adalah entitas dasar run-time dalam suatu sistem berorientasi objek. Masalah pemrograman dianalisis dalam hal objek dan sifat komunikasi antara mereka. Ketika program dieksekusi, objek berinteraksi satu sama lain dengan

mengirimkan pesan. Objek yang berbeda juga dapat berinteraksi satu sama lain tanpa mengetahui rincian data atau kode mereka.

Operator kondisi ?: adalah operator ternary. Berarti bahwa operator ini membawa tiga argumen yang membentuk suatu ekspresi bersyarat.

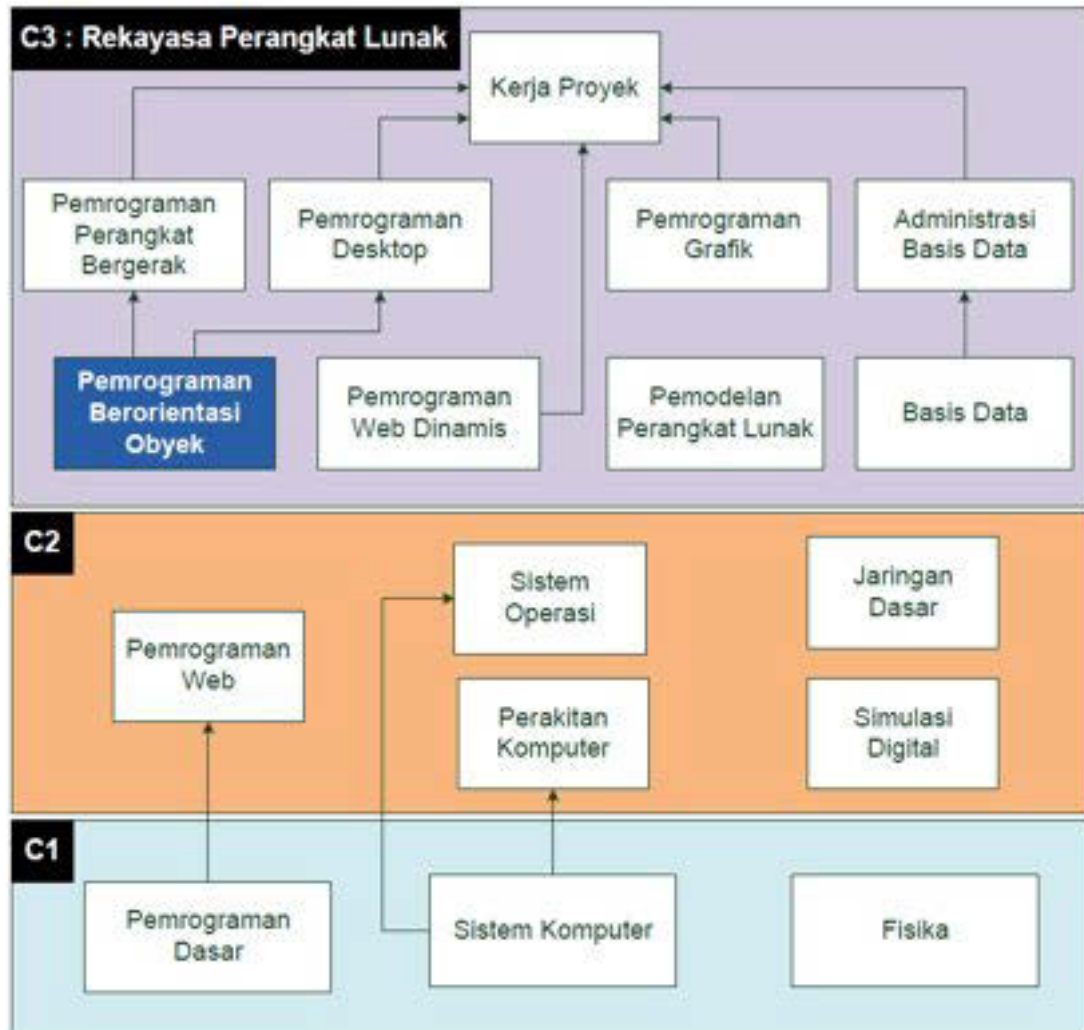
Operator precedence didefinisikan sebagai perintah yang dilakukan compiler ketika melakukan evaluasi terhadap operator, untuk mengajukan perintah dengan hasil yang tidak ambigu/ hasil yang jelas.

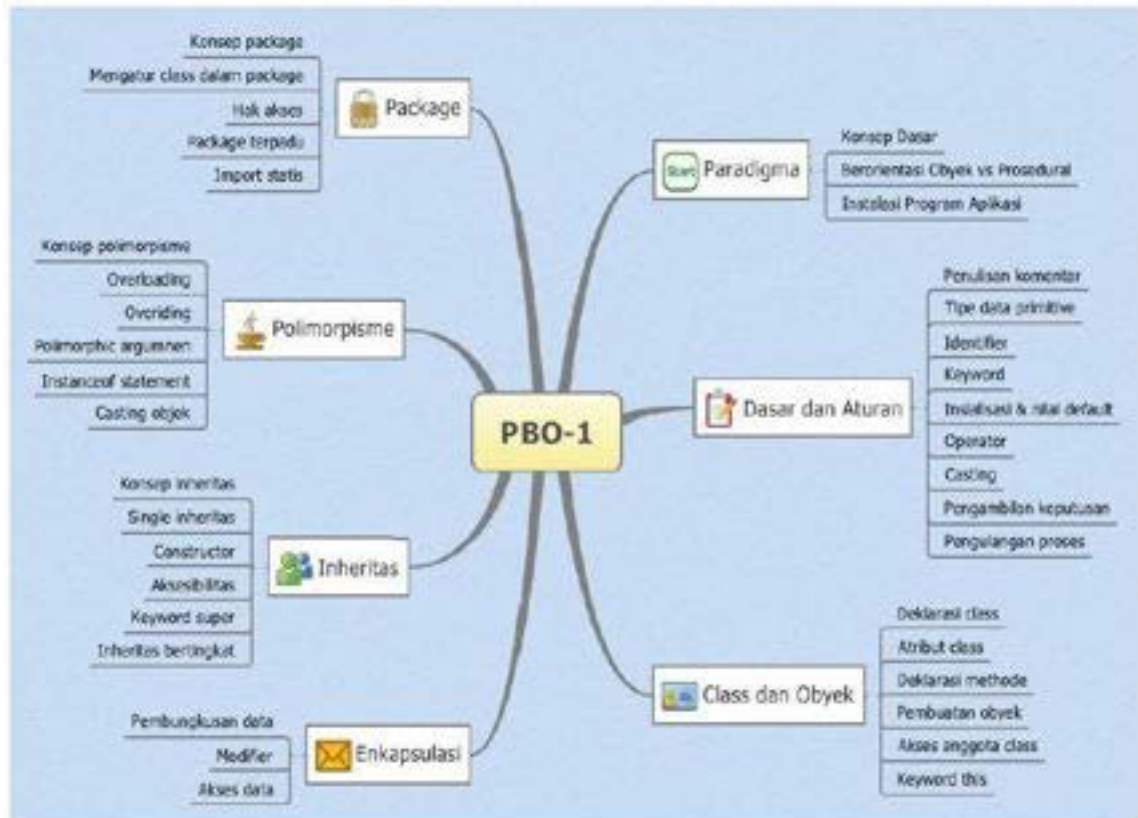
Pesan Passing adalah proses di mana sebuah objek mengirim data ke obyek lain atau meminta objek lain untuk memanggil sebuah metode . Message passing sesuai dengan " metode panggilan " .

Polimorfisme berarti kemampuan untuk mengambil lebih dari satu bentuk . Suatu operasi dapat menunjukkan perilaku yang berbeda dalam kasus yang berbeda . Perilaku tergantung pada jenis data yang digunakan dalam operasi.

Static adalah method yang dapat dipakai tanpa harus menginisialisasi suatu class (maksudnya tanpa menggunakan variabel terlebih dahulu).

While loop adalah pernyataan atau blok pernyataan yang diulang-ulang sampai mencapai kondisi yang cocok.

PETA KEDUDUKAN BUKU

Keterangan	
C1	Kelompok mata pelajaran Dasar Bidang Keahlian Teknologi Informasi dan Komunikasi
C2	Kelompok mata pelajaran Dasar Program Keahlian Teknik Komputer dan Informatika
C3	Kelompok mata pelajaran Paket Keahlian Rekayasa Perangkat Lunak
	Mata pelajaran Pemrograman Berorientasi Obyek Semester 1
	Mata pelajaran prasyarat

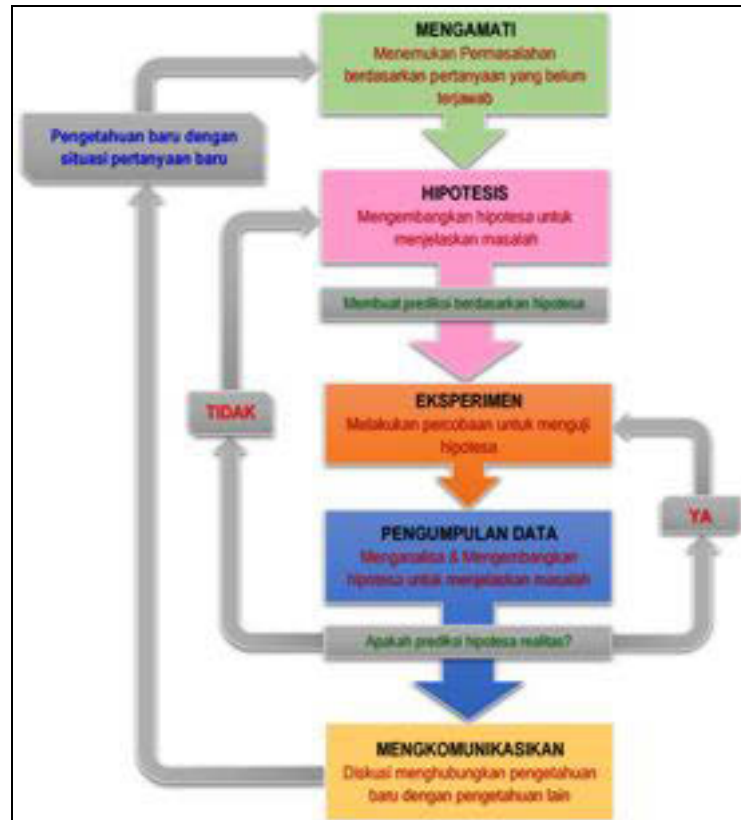
Peta Konsep : Pemrograman Berorientasi Obyek Kelas XI Semester 1

Keterangan	
KD 3.1- 4.1	Paradigma Pemrgraman Berorientasi Obyek
KD 3.2- 4.2	Dasar dan Aturan
KD 3.3- 4.3	Class dan Obyek
KD 3.4- 4.4	Enkapsulasi
KD 3.5- 4.5	Inheritas
KD 3.6- 4.6	Polimopisme
KD 3.7- 4.7	Package

BAB I PENDAHULUAN

A. Deskripsi

Pemrograman berorientasi objek (Inggris: *object-oriented programming* disingkat OOP) merupakan paradigma pemrograman yang berorientasikan kepada objek. Ini adalah jenis pemrograman di mana programmer mendefinisikan tidak hanya tipe data dari sebuah struktur data, tetapi juga jenis operasi (fungsi) yang dapat diterapkan pada struktur data. Dengan cara ini, struktur data menjadi objek yang meliputi data dan fungsi. Selain itu, pemrogram dapat membuat hubungan antara satu benda dan lainnya. Sebagai contoh, objek dapat mewarisi karakteristik dari objek lain.

Salah satu keuntungan utama dari teknik pemrograman berorientasi obyek atas teknik pemrograman prosedural adalah bahwa memungkinkan programmer untuk membuat modul yang tidak perlu diubah ketika sebuah jenis baru objek ditambahkan. Seorang pemrogram hanya dapat membuat objek baru yang mewarisi banyak fitur dari objek yang sudah ada. Hal ini membuat program object-oriented lebih mudah untuk memodifikasi.

Pembelajaran pemrograman berorientasi obyek ini menggunakan metode pendekatan saintifik. Dalam pendekatan ini praktikum atau eksperimen berbasis sains merupakan bidang pendekatan ilmiah dengan tujuan dan aturan khusus, dimana tujuan utamanya adalah untuk memberikan bekal ketrampilan yang kuat dengan disertai landasan teori yang realistis mengenai fenomena yang akan kita amati. Ketika suatu permasalahan yang hendak diamati memunculkan pertanyaan-pertanyaan yang tidak bisa terjawab, maka metode eksperimen ilmiah hendaknya dapat memberikan jawaban melalui proses yang logis. Proses-proses dalam pendekatan scientifik meliputi beberapa tahapan yaitu: mengamati, hipotesis atau menanya, mengasosiasikan atau eksperimen, mengumpulkan atau analisa data dan mengkomunikasikan. Proses belajar pendekatan eksperimen pada hakekatnya merupakan proses berfikir ilmiah untuk membuktikan hipotesis dengan logika berfikir.

Gambar 1. Diagram Proses Metode Saintifik-Eksperimen Ilmiah

B. Prasyarat

Untuk kelancaran pencapaian kompetensi dalam mata pelajaran pemrograman berorientasi obyek ini dibutuhkan beberapa peryaratan baik pengetahuan maupun ketrampilan dasar. Persyaratan tersebut antara lain ialah: Peserta didik telah menguasai mata pelajaran pemrograman dasar. Konsep dan algoritma pemrograman ini dibutuhkan untuk mendukung implementasi pemrograman berorientasi obyek. Disamping itu peserta didik mempunyai kompetensi dalam hal pemanfaatan teknologi informasi, seperti mengoperasikan hardware komputer dan mengoperasikan perangkat lunak aplikasi. Perangkat lunak aplikasi tersebut antar lain ialah pengolah data untuk menganalisis data hasil eksperimen, pengolah kata untuk membuat laporan dan aplikasi presentasi untuk mengkomunikasikan dan mempresentasikan hasil laporan.

C. Petunjuk Penggunaan

Buku pedoman siswa ini disusun berdasarkan kurikulum 2013 yang mempunyai ciri khas penggunaan metode saintifik. Buku ini terdiri dari dua bab

yaitu bab satu pendahuluan dan bab dua pembelajaran. Dalam bab pendahuluan beberapa yang harus dipelajari peserta didik adalah diskripsi mata pelajaran yang berisi informasi umum, rasionalisasi dan penggunaan metode scientifik. Selanjutnya pengetahuan tentang persyaratan, tujuan yang diharapkan, kompetensi inti dan dasar yang akan dicapai serta test kemampuan awal.

Bab dua menuntun peserta didik untuk memahami deskripsi umum tentang topik yang akan dipelajari dan rincian kegiatan belajar sesuai dengan kompetensi dan tujuan yang akan dicapai. Setiap kegiatan belajar terdiri dari tujuan dan uraian materi topik pembelajaran, tugas serta test formatif. Uraian pembelajaran berisi tentang diskripsi pemahaman topik materi untuk memenuhi kompetensi pengetahuan. Uraian pembelajaran juga menjelaskan deskripsi unjuk kerja atau langkah-langkah logis untuk memenuhi kompetensi skill.

Tugas yang harus dikerjakan oleh peserta didik dapat berupa tugas praktek, eksperimen atau pendalaman materi pembelajaran. Setiap tugas yang dilakukan melalui beberapa tahapan saintifik yaitu: 1) melakukan pengamatan setiap tahapan unjuk kerja; 2) melakukan praktek sesuai dengan unjuk kerja; 3) mengumpulkan data yang dihasilkan setiap tahapan; 4) menganalisa hasil data menggunakan analisa diskriptif; 5) mengasosiasikan beberapa pengetahuan dalam uraian materi pembelajaran untuk membentuk suatu kesimpulan; dan 6) mengkomunikasikan hasil dengan membuat laporan portofolio. Laporan tersebut merupakan tagihan yang akan dijadikan sebagai salah satu referensi penilaaian.

D. Tujuan Akhir

Setelah mempelajari uraian materi dalam bab pembelajaran dan kegiatan belajar diharapkan peserta didik dapat memiliki kompetensi sikap, pengetahuan dan keterampilan yang berkaitan dengan materi:

- ✓ Paradigma pemrograman berorientasi obyek
- ✓ Dasar dan aturan pemrograman berorientasi obyek
- ✓ Class dan Obyek
- ✓ Enkapsulasi data
- ✓ Pemawarisan
- ✓ Polimorpisme
- ✓ Package

E. Kompetensi Inti Dan Kompetensi Dasar

1. **Kompetensi Inti 1 :** Menghayati dan mengamalkan ajaran agama yang dianutnya.

   Kompetensi Dasar :

   1.1. Memahami nilai-nilai keimanan dengan menyadari hubungan keteraturan dan kompleksitas alam dan jagad raya terhadap kebesaran Tuhan yang menciptakannya

   1.2. Mendeskripsikan kebesaran Tuhan yang menciptakan berbagai sumber energi di alam

   1.3. Mengamalkan nilai-nilai keimanan sesuai dengan ajaran agama dalam kehidupan sehari-hari.

2. **Kompetensi Inti 2:** Menghayati dan Mengamalkan perilaku jujur, disiplin, tanggung jawab, peduli (gotong royong, kerjasama, toleran, damai), santun, responsif dan proaktif dan menunjukan sikap sebagai bagian dari solusi atas berbagai permasalahan dalam berinteraksi secara efektif dengan lingkungan sosial dan alam serta dalam menempatkan diri sebagai cerminan bangsa dalam menempatkan diri sebagai cerminan bangsa dalam pergaulan dunia.

   Kompetensi Dasar:

   2.1.Menunjukkan perilaku ilmiah (memiliki rasa ingin tahu; objektif; jujur; teliti; cermat; tekun; hati-hati; bertanggung jawab; terbuka; kritis; kreatif; inovatif dan peduli lingkungan) dalam aktivitas sehari-hari sebagai wujud implementasi sikap dalam melakukan percobaan dan berdiskusi

   2.2.Menghargai kerja individu dan kelompok dalam aktivitas sehari-hari sebagai wujud implementasi melaksanakan percobaan dan melaporkan hasil percobaan.

3. **Kompetensi Inti 3:** Memahami, menerapkan dan menganalisis pengetahuan faktual, konseptual dan prosedural berdasarkan rasa ingin tahunya tentang ilmu pengetahuan, teknologi, seni, budaya, dan humaniora dalam wawasan kemanusiaan, kebangsaan, kenegaraan, dan peradaban terkait penyebab fenomena dan kejadian dalam bidang kerja yang spesifik untuk memecahkan masalah.

   Kompetensi Dasar:

   3.1. Memahami konsep pemrograman berorientasi obyek

3.2. Memahami dasar-dasar dan aturan pemrograman berorientasi obyek
3.3. Memahami konsep class dan obyek
3.4. Memahami konsep enkapsulasi dalam melindungi data dan informasi
3.5. Memahami konsep pewarisan
3.6. Memahami konsep polimorphisme
3.7. Menerapkan penggunaan pakage dalam aplikasi

4. **Kompetensi Inti 4:** Mengolah, menalar, dan menyaji dalam ranah konkret dan ranah abstrak terkait dengan pengembangan dari yang dipelajarinya di sekolah secara mandiri, dan mampu melaksanakan tugas spesifik dibawah pengawasan langsung.

   Kompetensi Dasar:

   4.1 Menyajikan konsep pemrograman berorientasi obyek
   4.2 Menyajikan aturan dan dasar-dasar pemrograman berorientasi obyek
   4.3 Menyajikan class dengan memberikan atribut dan metode
   4.4 Menyajikan perlindungan data dan informasi melalui mekanisme enkapsulas
   4.5 Mengolah hubungan antara class dengan pola pewarisan
   4.6 Menyajikan konsep polimorphisme dengan overloading dan overiding
   4.7 Menyajikan aplikasi melalui pengelompokan class dalam pakage

F. Cek Kemampuan Awal

1. Jelaskan perbedaan permrograman prosedural dengan pemrograman berorientasi obyek!
2. Jelaskan keuntungan pemrograman berorientasi obyek!
3. Jelaskan secara singkat proses kompilasi dan menjalankan program aplikasi berorientasi obyek!
4. Jelaskan secara singkat dan berikan contoh pengertian class, atribut dan metode!
5. Jelaskan pengertian pembungkusan data dengan konsep enkapsulasi!
6. Jelaskan secara singkat konsep pewarisan dalam pemrograman berorientasi obyek!
7. Jelaskan secara singkat konsep polimorpisme dalam pemrograman berorientasi obyek!
8. Jelaskan cara pembuatan package dan sebutkan keuntungannya !

BAB II PEMBELAJARAN

A. Deskripsi

Dalam bab 1 ini akan menjelaskan dan menyajikan konsep pemrograman berorientasi obyek yang terdiri dari 2 kegiatan belajar. Kegiatan belajar 1 akan memahamkan anda tentang paradigma pemrograman berorientasi obyek dan menganalisis perbedaan pemrograman procedural dan pemrograman perorientasi obyek. Kegiatan belajar 2 meliputi penjelasan alur kerja perangkat lunak berorientasi obyek dan melakukan instalasi perangkat lunak. Setiap kegiatan belajar disertai dengan tujuan pembelajaran yang akan dicapai dalam 1 kali tatap muka, uraian materi, tes formatif untuk menguji kompetensi pengetahuan anda, dan tugas atau praktikum(individu dan kelompok) untuk menguji kompetensi keterampilan anda.

B. Kegiatan Belajar

1. Kegiatan Belajar 1 : Mengenal Pemrograman Berorientasi Obyek

a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar satu ini siswa diharapkan dapat:

1) Memahami paradigma pemrograman berorientasi obyek
2) Menganalisis perbandingan pemrograman prosedural dan pemrograman berorientasi obyek

b. Uraian Materi

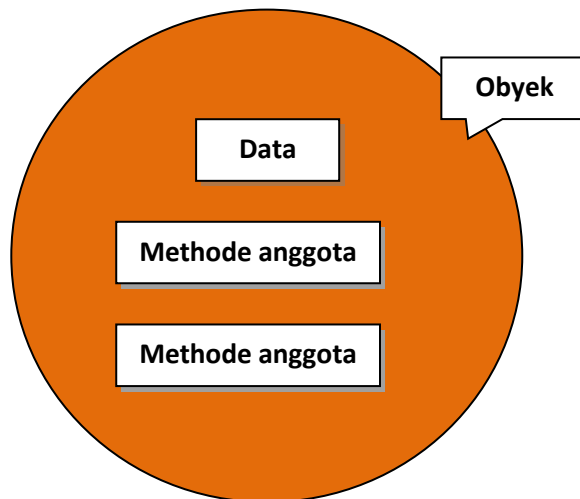
1) Paradigma Pemrograman Berorientasi Obyek

Ide dasar pada bahasa berorientasi obyek (POB) adalah mengkombinasikan data dan fungsi untuk mengakses data menjadi sebuah kesatuan unit. Unit ini di kenal dengan nama **obyek.** Obyek sebenarnya mencerminkan pola kerja manusia dalam kehidupan kerja sehari-hari. Sebuah obyek dapat diibaratkan sebagai departemen di dalam sebuah perusahaan bisnis.

Contoh departemen:

- ✓ Penjualan
- ✓ Akuntan
- ✓ Personalia

Pembagian departemen dalam perusahaan merupakan upaya untuk memudahkan pengoperasian perusahaan. Sebagai gambaran, jika anda seorang menejer penjualan di Kantor Pusat ingin mengetahui data personalia *salesmen* di suatu kantor cabang, apa yang anda lakukan? Langkah yang anda tempuh pasti tidak datang secara langsung ke ruang personalia dan mencari data pada berkas-berkas yang ada pada departemen personalia sesuai yang anda butuhkan. Masalah bagaimana dan siapa yang mencarikan laporan yang diperlukan bukanlah menjadi urusan anda. Analogi dengan hal ini, kalau seseorang bermaksud menggunakan obyek, ia cukup mengirim suatu pesan ke obyek dan obyek itu sendiri yang akan menanganinya.

Bisa dibayangkan, betapa repotnya anda kalau anda sebagai manejer penjualan harus mencari sendiri berkas-berkas yang ada pada departemen personalia. Barangkali anda malah bakal mengobrak-abrik berkas-berkas yang sudah tersusun rapi. Kejadian semacam inilah yang dihindari pada konsep pemrogaman berorientasi obyek. Sebuah gambaran tentang obyek yang berisi data dan fungsi yang memanipulasi data dapat dilihat pada Gambar 1.

Gambar 1. Data dan fungsi pemanipulasian data pada suatu obyek

- **Obyek**

Object adalah gabungan antara beberapa data dan fungsi yang masing-masing bekerja bersama-sama dan tidak dapat dipisahkan. Gabungan dari data dan fungsi tersebut akan membentuk suatu object-object yang aktif. Dari kumpulan beberapa *object* yang sama akan membentuk struktur baru yang disebut class.

Konsep dasar object (*object Oriented)* meliputi tiga hal:

- ✓ Is Identical (*because Object has own unique ID*), yaitu object tersebut mempunyai identitas tersendiri dapat dibedakan dengan yang lain

- ✓ Has Behavior (*because Object has Method*), yaitu object itu mempunyai prilaku atau sifat-sifat yang khusus
- ✓ Has State (*because Object has instance parameter*), object mempunyai ukuran yang baku.

Dalam suatu sistem yang komplek seperti dalam proyek manajemen sering kita jumpai objek-objek suatu kelas mempunyai relasi/hubungan dengan object-object dikelas yang lain. Secara umum relasi object dapat dibedakan menjadi tiga dasar hubungan yaitu :

- ✓ Is-a (Generalization, Realization: Inheritance).
- ✓ Has-a (Association).
- ✓ Others (Association , Dependency)

➢ **Karakteristik Obyek**

Untuk lebih jelasnya karakteristik objek tersebut dijelaskan sebagai berikut :

Gambar 2.Contoh obyek

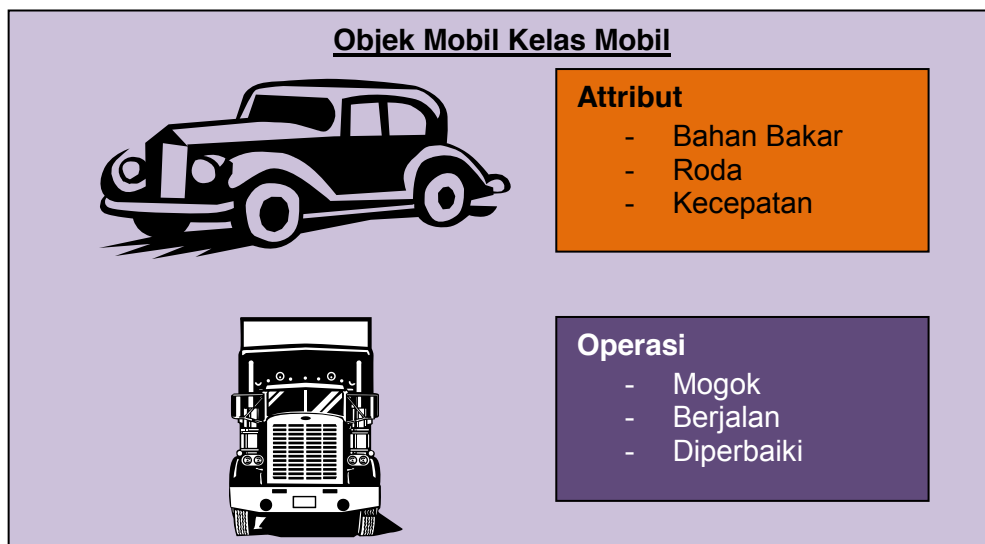
- ❖ Identitas berarti data diukur mempunyai nilai tertentu yang membedakan entitas dan inilah yang disebut objek

Suatu contoh: Singa merupakan obyek dari binatang buas, Manusia merupakan obyek dari makhluk hidup ciptaan Allah SWT, keyboard merupakan objek dari perangkat keras komputer, mobil merupakan objek dari alat transportasi. Setiap objek mempunyai sifat yang melekat pada identitasnya, sehingga dua objek dapat berbeda walaupun bila semua nilai atributnya identik. Lebih jelasnya lihat gambar 3.

Objek didunia nyata sangatlah sederhana, tetapi dalam pemrograman mempunyai penanganan yang unik dan tidak sederhana di dunia nyata. Penanganannya bisa saja dinyatakan dengan beberapa cara, seperti alamat, indeks, dari array atau nilai unik dari atribut. Referensi objek seragam dan independen dari sisi objek, memperbolehkan campuran kumpulan dari objek yang dibuat, seperti file dalam direktori yang berisi file dan subdirektori.

- ❖ Klasifikasi berarti suatu kegiatan mengumpulkan data (atribut) dan prilaku (operasi) yang mempunyai struktur data sama ke dalam satu grup yang

disebut kelas. Sesuai dengan contoh di atas maka hewan buas, makhluk hidup, komputer, dan alat transportasi adalah contoh dari kelas.

Kelas menunjukkan abstraksi yang menjelaskan sifat penting pada suatu aplikasi dan mengabaikan yang lain. Setiap kelas menunjukkan suatu kumpulan *infinite* yang mungkin dari objek. Suatu obyek dapat dikatakan sebagai instans dari kelas. Setiap instans dari kelas mempunyai nilai individu untuk setiap nama atribut dan operasi, tetapi memiliki bersama atribut dan operasi dengan instans lain dalam kelas. Gambar di bawah ini menunjukkan dua kelas dengan beberapa instans yang berhubungan dengannya.

Gambar 3. Contoh obyek dengan properti

Dalam dunia nyata, suatu operasi adalah suatu abstrak dari analogi prilaku terhadap obyek-obyek yang berbeda. Setiap objek mengetahui bagaimana melakukan operasinya. Dalam bahasa pemrograman berorientasi objek, secara otomatis bahasa akan memilih metode yang tepat untuk menjalankan operasinya berdasarkan nama dimana dilakukan operasi terhadapnya. Pengguna dari operasi tidak perlu khawatir berapa banyak metoda yang terdapat dalam implementasi. Kelas baru dapat ditambahkan tanpa mengubah *code* yang sudah ada, melengkapi metoda adalah melengkapi operasi yang dapat dilakukan terhadap kelas baru.

Perbandingan Pemrograman Berorientasi Obyek dengan Pemrograman Terstruktur

Paradigma bahasa pemrograman memberikan model untuk programmer dalam menulis listing program. Paradigma perbedaan dalam bahasa pemrograman sebagai berikut :

Gambar 4. Bahasa pemrograman

- ❖ Pemrograman tidak terstruktur atau *Programming Monolithic*
- ❖ Pemrograman prosedural
- ❖ Pemrograman struktural
- ❖ Pemrograman Berorientasi Objek

❖ Pemrograman tidak terstruktur atau *Programming Monolithic* :

- ✓ Seluruh permasalahan ini diselesaikan sebagai blok tunggal.
- ✓ Semua data bersifat global dan tidak ada keamanan.
- ✓ Perintah melompat diperbolehkan `jump` dan banyak menggunakan perintah `go to`
- ✓ Cocok untuk permasalah kecil.
- ✓ Sulit untuk melacak kesalahan program

Contoh bahasa pemrograman yang termasuk dalam *Programming Monolithic* adalah Assembly Language, BASIC.

Gambar 5.Diagram bahasa pemrograman

❖ **Pemrograman prosedural**

Masalah yang diberikan dibagi dalam beberapa sub masalah tergantung pada fungsinya.

Masalah disebut prosedur atau Metode.

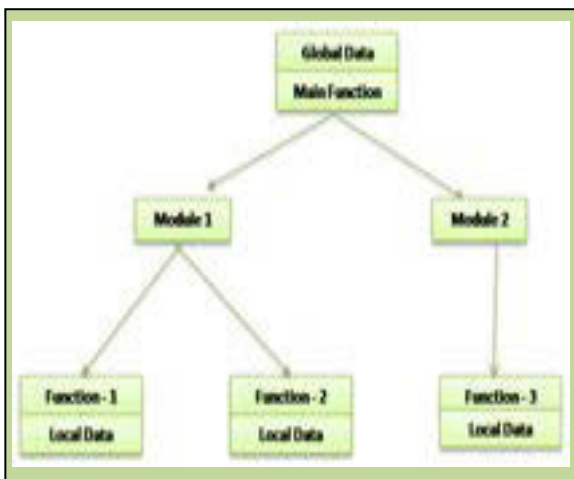
Prosedur apapun dapat dipanggil pada setiap saat selama pelaksanaan program.

Program ini memiliki variabel global dan lokal.

Fitur Pemrograman berorientasi prosedur:

- ✓ Program besar yang terbagi dalam fungsi kecil atau Prosedur.
- ✓ Menggunakan Pendekatan pemrograman *Top-Down*.
- ✓ Data bergerak bebas dari satu fungsi ke yang lain.
- ✓ Sebagian besar fungsi berbagi data umum.
- ✓ Penekanan diberikan untuk algoritma.

Kekurangan:

- ✓ Sangat sulit mengidentifikasi data yang digunakan oleh yang berfungsi.
- ✓ Sulit untuk melacak kesalahan program

Gambar 6. Diagram bahasa pemrograman

❖ **Pemrograman Terstruktur**

Program ini dibagi menjadi modul dan modul tersebut kemudian dibagi menjadi fungsi.

Penggunaan Pernyataan `go to` dihapus atau dikurangi.

Setiap modul dapat bekerja independen satu sama lain.

❖ **Pemrograman berorientasi obyek**

- ✓ Program ini dibagi menjadi jumlah unit kecil yang disebut Object. Data dan fungsi merupakan properti objek.
- ✓ Data dari objek hanya dapat diakses oleh fungsi yang terkait dengan objek tersebut.
- ✓ Fungsi satu objek dapat mengakses fungsi objek lain.

Fitur pemrograman berorientasi obyek

- ✓ Penekanan diberikan pada data daripada prosedur.
- ✓ Masalah dibagi menjadi obyek.
- ✓ Struktur data dirancang sedemikian rupa sehingga mereka mengatur objek.
- ✓ Data dan fungsi yang diikat bersama-sama.
- ✓ Penyembunyian data adalah mungkin.

- ✓ Data baru dan fungsi dapat dengan mudah dibuat.
- ✓ Obyek dapat berkomunikasi satu sama lain dengan menggunakan fungsi.
- ✓ Pendekatan bottom-up yang digunakan dalam membuat program

Perbedaan antara Pemrograman Berorientasi Terstruktur dan Obyek

Pemrograman Terstruktur	Pemrograman Berorientasi Obyek
Pendekatan *top-down*	Pendekatan *bottom-up* yang diikuti.
Fokus adalah pada algoritma dan kontrol aliran.	Fokus pada model obyek.
Program dibagi menjadi beberapa sub modul atau fungsi atau prosedur.	Program ini diselenggarakan dengan memiliki sejumlah kelas dan objek.
Fungsi yang independen satu sama lain.	Setiap kelas berhubungan secara hirarkis.
Tidak ada penerima yang ditunjuk dalam panggilan fungsi.	Ada penerima yang ditunjuk untuk setiap lewat pesan.
Data dan fungsi sebagai dua entitas yang terpisah *Views*.	Data dan fungsi sebagai satu kesatuan pandangan.
Pemeliharaan mahal.	Pemeliharaan relatif lebih murah.
Reuse Software tidak mungkin.	Membantu dalam penggunaan kembali perangkat lunak.
Fungsi panggilan digunakan.	Message passing digunakan.
Fungsi abstraksi digunakan.	Data abstaction digunakan.
Algoritma diberikan penting.	Data diberikan penting.
Solution adalah solusi spesifik-domain.	*Solution* adalah spesifik masalah domain.
Tidak ada enkapsulasi. Data dan fungsi yang terpisah	Enkapsulasi paket kode dan data sama sekali. Data dan fungsi disatukan dalam satu kesatuan.
Hubungan antara programmer dan program ditekankan.	Hubungan antara programmer dan pengguna ditekankan.
Teknik data-driven digunakan.	Didorong oleh delegasi tanggung jawab.

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sedangkan untuk pemrograman terstruktur, menggunakan prosedur/tata cara yang teratur untuk mengoperasikan data struktur. Untuk tata nama, keduanya pun memiliki tatanan yang sama walaupun memiliki pengertian tersendiri.

Object oriented menggunakan "method" sedangkan terstruktur menggunakan "function". Bila di OOP sering didengar mengenai "objects" maka di terstruktur kita mengenalnya dengan "modules". Begitu pula halnya dengan "message" pada OO dan "argument" pada terstruktur. "attribute" pada OO juga memiliki tatanan nama yang sepadan dengan "variabel" pada pemrograman terstruktur.

Pemrograman prosedural akan dikatakan lebih baik apabila dalam segala situasi melibatkan kompleksitas moderat atau yang memerlukan signifikan kemudahan *maintainability*. Manfaat yang dirasakan dalam penggunaan pemrograman prosedural adalah kemampuan kembali menggunakan kode yang sama tanpa menggunakan kode yang berbeda ataupun mengkopinya kembali. Dengan menggunakan "*goto*", memudahkan programmer melacak kumpulan data sehingga menghindarkan pemrograman terstruktur menjadi seperti spagethii code.

Pemrograman berorientasikan objek dikatakan lebih baik apabila model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

c. Rangkuman

Pengembangan berorientasi objek merupakan cara pikir baru tentang perangkat lunak berdasarkan abstraksi yang terdapat dalam dunia nyata. Dalam konteks pengembangan menunjuk pada bagian awal dari siklus hidup pengembangan sistem, yaitu survei, analisis, desain, implementasi, dan pemeliharaan sistem. Hal yang lebih penting dalam pengembangan berorientasi objek adalah konsep mengindentifikasi dan mengorganisasi domain aplikasi dibandingkan dengan fokus penggunaan bahasa pemrograman, berorientasi objek atau tidak.

Object adalah gabungan antara beberapa data dan fungsi yang masing-masing bekerja bersama-sama dan tidak dapat dipisahkan. Gabungan dari data dan fungsi tersebut akan membentuk suatu *object-object* yang aktif. Dari kumpulan beberapa *object* yang sama akan membentuk struktur baru yang disebut *class*.

Pemrograman berorientasi obyek (Inggris: *object-oriented programming* disingkat OOP) merupakan paradigma pemrograman yang berorientasikan kepada obyek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau obyek-obyek. Bandingkan dengan logika pemrograman terstruktur. Setiap obyek dapat menerima pesan, memproses data, dan mengirim pesan ke obyek lainnya.

Pemrograman terstruktur adalah suatu proses untuk mengimplementasikan urutan langkah untuk menyelesaikan suatu masalah dalam bentuk program. Selain pengertian diatas Pemrograman Terstruktur adalah suatu aktivitas pemrograman dengan memperhatikan urutan langkah-langkah perintah secara sistematis, logis, dan tersusun berdasarkan algoritma yang sederhana dan mudah dipahami. Prinsip dari pemrograman terstruktur adalah Jika suatu proses telah sampai pada suatu titik/langkah tertentu ,maka proses selanjutnya tidak boleh mengeksekusi langkah sebelumnya/kembali lagi ke baris sebelumnya, kecuali pada langkah–langkah untuk proses berulang (*Loop*).

d. Tugas

Tugas 1

Tentukan atribut dan operasi dari obyek Komputer!

❖ Mengamati benda dan obyek

1. Buatlah kelompok dengan anggota 3 – 4 orang.
2. Amatilah obyek disekitar anda.
3. Sebutkan ciri-ciri atau atribut dari obyek yang Anda amati.
4. Sebutkan fungsi yang merupakan operasi relasi dari ciri-ciri yang sudah teridentifikasi.
5. Deskripsikan setiap benda tersebut seperti gambar berikut.

6. Buat laporan dan diskusikan dengan teman sekelompok.

❖ Bandingkan dan Simpulkan

Bandingkan hasil pendiskripsian suatu benda yang meliputi (nama benda, atribut, operasi) dari hasil kerja kelompok Anda dengan kelompok lain.

Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama.

Tugas 2

Buatlah kesimpulan dari perbandingan pemrograman prosedural dan pemrograman berorientasi obyek.

❖ Mengamati Bahasa Pemrograman

1. Buatlah kelompok dengan anggota 3 – 4 orang
2. Amatilah dan deskripsikan perbandingan pemrograman prosedural dan pemrograman berorientasi obyek
3. Tunjukkanlah perbedaan yang mendasar dari 2 bahasa pemrograman yang sudah anda kenal
4. Sebutkan persamaan fungsi atau kegunaan yang dapat terdapat dalam 2 bahasa pemrograman yang sudah Anda ketahui
5. Buatlah tabel perbedaan dan persamaan yang Anda dapatkan diskusikan dengan teman sekelompok.

No	Bahasa Pemrograman	Sistem Operasi	File Extensi	Jenis Publikasi
1.				
2.				

❖ Bandingkan dan Simpulkan

Bandingkan hasil tabel pengamamatan terhadap 2 bahasa pemrograman yang sudah anda kenal dari hasil kerja kelompok anda dengan kelompok lain.

Berdasarkan hasil pengamatam tersebut hal penting apa yang harus dirumuskan secara bersama

e. Test Formatif

Dalam test ini setiap Anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Ilustrasikan dalam dunia nyata apa yang disebut :
   a. Objek
   b. Atribut
   c. Methode
   d. Kelas
2. Sebutkan minimal 5 atribut dan minimal 3 method yang melekat pada diri anda ?
3. Sebutkan paradigma lain dalam bahasa pemrograman selain paradigma berorientasi objek, bandingkan dan sebutkan masing-masing kekurangan dan kelemahannya ?
4. Mengapa saat ini metodologi berorientasi objek berkembang lebih pesat dibandingkan dengan metode-metode yang lain dalam bahasa pemrograman

f. Lembar Jawaban Test Formatif (LJ)

LJ- 01 : Ilustrasi dalam dunia nyata:

a) Objek...
....
....
....
b) Atribut ……..
....
....
....
c) Methode..
....
....
....
d) Kelas……...
....
....

g. Lembar Kerja Siswa

2. Kegiatan Belajar 2 : Perangkat Lunak Pemrograman Berorientasi Obyek

a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 2 ini siswa diharapkan dapat :

1) Memahami alur kerja perangkat lunak pemrograman berorientasi obyek
2) Menyajikan perangkat lunak pemrograman berorientasi obyek

b. Uraian Materi

1) Alur kerja perangkat lunak pemrograman berorientasi obyek

Mengapa memilih Java sebagai perangkat lunak pemrograman berorientasi obyek?

Java diciptakan oleh suatu tim yang dipimpin oleh Patrick Naughton dan James Gosling dalam suatu proyek dari *Sun Micro System* yang memiliki kode Green dengan tujuan untuk menghasilkan bahasa komputer sederhana yang dapat dijalankan di peralatan sederhana dengan tidak terikat pada arsitektur tertentu. Mulanya disebut OAK, tetapi karena OAK sendiri merupakan nama dari bahasa pemrograman komputer yang sudah ada. Maka *Sun* mengubahnya menjadi Java.

Sun kemudian meluncurkan *browser* dari Java yang disebut Hot Java yang mampu menjalankan applet. Setelah itu teknologi Java diadopsi oleh *Netscape* yang memungkinkan program Java dijalankan di *browser Netscape* yang kemudian diikuti Internet Explorer. Karena keunikanya dan kelebihanya, teknologi Java mulai menarik banyak vendor seperti IBM, Symantec, Inprise, dan lain-lain. Sun merilis versi awal Java secara resmi pada awal tahun 1996 yang kemudian terus berkembang hingga muncul JDK 1.1, kemudian JDK 1.2 yang mulai disebut sebagai versi Java2 karena banyak mengandung peningkatan dan perbaikan. Perubahan utama adalah adanya Swing yang merupakan teknologi GUI (*Graphical User Interface*) yang mampu menghasilkan window yang portabel. Dan pada tahun 1998–1999 lahirlah teknologi J2EE. Diawali dengan servlet dan EJB kemudian diikuti JSP. Java juga menjadi lebih cepat populer di lingkungan *server side* dikarenakan kelebihanya di lingkungan.

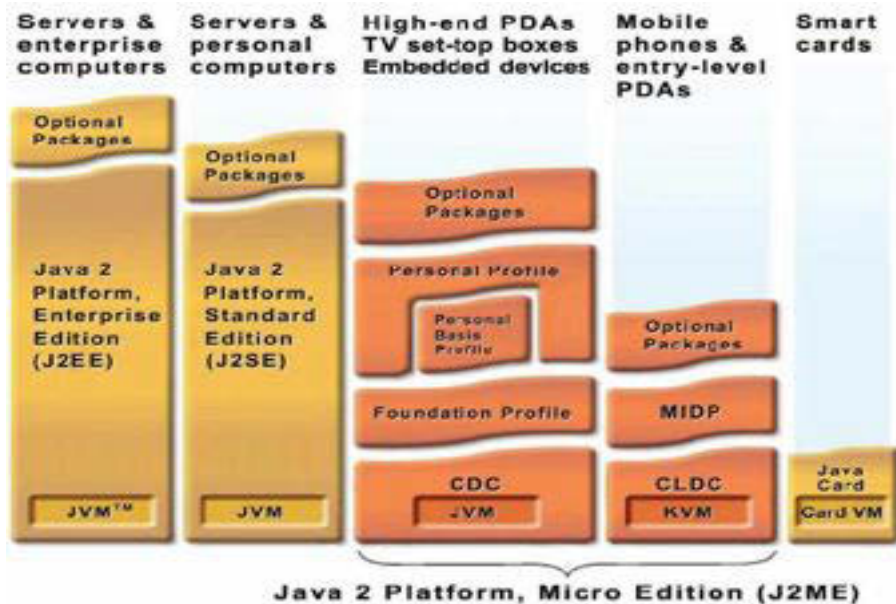
2) Arsitektur teknologi java

Java adalah suatu teknologi di dunia *software* komputer, yang merupakan suatu bahasa pemrograman, dan sekaligus suatu platform. Sebagai bahasa pemrograman, Java dikenal sebagai bahasa pemrograman tingkat tinggi. Java mudah dipelajari, terutama bagi programmer yang telah mengenal C/C++. Java merupakan bahasa pemrograman berorientasi objek yang merupakan paradigma pemrograman masa depan. Sebagai bahasa pemrograman Java dirancang menjadi handal dan aman. Java juga dirancang agar dapat dijalankan di semua platform. Dan juga dirancang untuk menghasilkan aplikasi–aplikasi dengan performansi yang terbaik, seperti aplikasi *database* Oracle 8i/9i yang *core*-nya dibangun menggunakan bahasa pemrograman Java. Sedangkan Java bersifat *neutral architecture,* karena Java *Compiler* yang digunakan untuk mengkompilasi kode program Java dirancang untuk menghasilkan kode yang netral terhadap semua arsitektur perangkat keras yang disebut sebagai *Java Bytecode*.

Sebagai sebuah platform, Java terdiri atas dua bagian utama, yaitu:

- Java Virtual Machine (JVM).
- Java Application Programming Interface (JavaAPI).

Sun membagi arsitektur Java membagi tiga bagian, yaitu:

- Enterprise Java (J2EE) untuk aplikasi berbasis web, aplikasi sistem tersebar dengan beraneka ragam klien dengan kompleksitas yang tinggi. Merupakan superset dari Standar Java
- Standar Java (J2SE), ini adalah yang biasa dikenal sebagai bahasa Java.
- Micro Java (J2ME) merupakan subset dari J2SE dan salah satu aplikasinya yang banyak dipakai adalah untuk wireless device / mobile device.

Gambar 7. Arsitektur Teknologi Java

1. Java API

Beberapa fitur yang ditawarkan Java API antara lain sebagai berikut:

a. **Applet**

Program Java yang dapat berjalan di atas browser, yang dapat membuat halaman HTML lebih dinamis dan menarik.

b. **Java Networking**

Sekumpulan API (*Application Programming Interface*) yang menyediakan fungsi–fungsi untuk aplikasi–aplikasi jaringan, seperti penyediaan akses untuk TCP, UDP, IP Adrress dan URL. Tetapi Java Networking tidak menyediakan akses untuk ICMP dikarenakan alasan sekuriti dan pada kondisi umum hanya administrator (root) yang bisa memanfaatkan protokol ICMP.

c. **JavaDatabase Connectivity (JDBC)**

JDBC menyediakan sekumpulan API yang dapat digunakan untuk mengakses database seperti Oracle, MySQL, PostgreSQL, Microsoft SQL Server.

d. **Java Security**

Java Security menyediakan sekumpulan API untuk mengatur security dari aplikasi Java baik secara *high level* atau *low level*, seperti *public/private key management* dan *certificates*.

e. JavaSwing

Java Swing menyediakan sekumpulan API untuk membangun aplikasi–aplikasi GUI (*Graphical User Interface*) dan model GUI yang diinginkan bisa bermacam–macam, bisa model Java, model Motif/CDE atau model yang *dependent* terhadap platform yang digunakan.

f. Java RMI

Java RMI menyediakan sekumpulan API untuk membangun aplikasi–aplikasi Java yang mirip dengan model RPC (*Remote Procedure Call*) jadi object-object Java bisa di call secara remote pada jaringan komputer.

g. Java2D/3D

Java 2D/3D menyediakan sekumpulan API untuk membangun grafik–grafik 2D/3D yang menarik dan juga akses ke printer.

h. Java Server Pages

Berkembang dari Java Servlet yang digunakan untuk menggantikan aplikasi–aplikasi CGI, JSP (*Java Server Pages*) yang mirip ASP dan PHP merupakan alternatif terbaik untuk solusi aplikasi Internet.

i. JNI (Java Native Interface)

JNI menyediakan sekumpulan API yang digunakan untuk mengakses fungsi – fungsi pada library (*.dll atau *.so) yang dibuat dengan bahasa pemrograman yang lain seperti C, C++, dan Basic.

j. JavaSound

Java Sound menyediakan sekumpulan API untuk manipulasi sound.

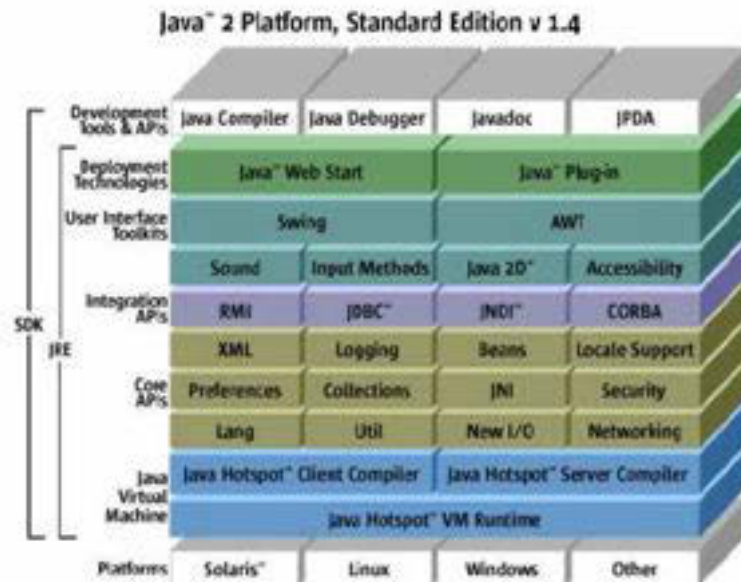
k. Java IDL + CORBA

Java IDL (*Interface Definition Language*) menyediakan dukungan Java untuk implementasi CORBA (*Common Object Request Broker*) yang merupakan model *distributed-Object* untuk solusi aplikasi besar di dunia *networking*.

l. JavaCard

Java Card utamanya digunakan untuk aplikasi–aplikasi pada *smart card*, yang sederhana wujudnya seperti SIM *Card* pada handphone.

m. JTAPI (Java Telephony API)

Java Telepony API menyediakan sekumpulan API untuk memanfaatkan *devices–devices telepony*, sehingga akan cocok untuk aplikasi–aplikasi CTI (*Computer Telephony Integration*) yang dibutuhkan seperti ACD (*Automatic Call Distribution*), PC- PBX .

Gambar 8. J2 SE Standard Edition

Berdasarkan *white paper* resmi dari SUN, Java memiliki karakteristik sebagai berikut :

- ✓ **Sederhana**

  Bahasa pemrograman Java menggunakan sintaks mirip dengan C++ namun sintaks pada Java telah banyak diperbaiki terutama menghilangkan penggunaan pointer yang rumit dan *multiple inheritance*. Java juga menggunakan *automatic memory allocation* dan *memory garbage collection*.

- ✓ **Berorientasi objek (*ObjectOriented*)**

  Java mengunakan pemrograman berorientasi objek yang membuat program dapat dibuat secara modular dan dapat dipergunakan kembali. Pemrograman berorientasi objek memodelkan dunia nyata ke dalam objek dan melakukan interaksi antar objek-objek tersebut.

- ✓ **Dapat didistribusi dengan mudah**

  Java dibuat untuk membuat aplikasi terdistribusi secara mudah dengan adanya *libraries networking* yang terintegrasi pada Java.

- ✓ **Interpreter**

  Program Java dijalankan menggunakan interpreter yaitu *Java Virtual Machine* (JVM). Hal ini menyebabkan *sourcecode* Java yang telah dikompilasi menjadi Java *bytecodes* dapat dijalankan pada plat form yang berbeda-beda.

- **Robust**

  Java mempunyai reliabilitas yang tinggi. Compiler pada Java mempunyai kemampuan mendeteksi error secara lebih teliti dibandingkan bahasa pemrograman lain. Java mempunyai *runtime Exception handling* untuk membantu mengatasi error pada pemrograman.

- **Aman**

  Sebagai bahasa pemrograman untuk aplikasi internet dan terdistribusi, Java memiliki beberapa mekanisme keamanan untuk menjaga aplikasi tidak digunakan untuk merusak *system computer* yang menjalankan aplikasi tersebut.

- **ArchitectureNeutral**

  Program Java merupakan *platform independent*. Program cukup mempunyai satu buah versi yang dapat dijalankan pada platform yang berbeda dengan *Java VirtualMachine*.

- **Portable**

  Source code maupun program Java dapat dengan mudah dibawa ke p latform yang berbeda-beda tanpa harus dikompilasi ulang.

- **Performance**

  Performance pada Java sering dikatakan kurang tinggi.Namun performance Java dapat ditingkatkan menggunakan kompilasi Java lain seperti buatan Inprise, Microsoft ataupun Symantec yang menggunakan *Just In Time Compilers* (JIT).

- **Multithreaded**

  Java mempunyai kemampuan untuk membuat suatu program yang dapat melakukan beberapa pekerjaan secara sekaligus dan simultan.

- **Dinamis**

  Java didesain untuk dapat dijalankan pada lingkungan yang dinamis. Perubahan pada suatu *class* dengan menambahkan *properties* ataupun *method* dapat dilakukan tanpa menggangu program yang menggunakan *class* tersebut.

Sebagai sebuah bahasa pemrograman, Java dapat membuat seluruh bentuk aplikasi, *desktop, web* dan lainnya, sebagaimana dibuat dengan menggunakan bahasa pemrograman konvensional yang lain.

Java adalah bahasa pemrograman yang berorientasi objek (OOP) dan dapat

dijalankan pada berbagai *platform* system operasi. Perkembangan Java tidak hanya terfokus pada satu sistem operasi, tetapi dikembangkan untuk berbagai sistem operasi dan bersifat *open source.*

Sebagian fitur dari Java antara lain:

- ✓ ***Java Virtual Machine (JVM)***

  JVM adalah sebuah mesin imajiner (maya) yang bekerja dengan menyerupai aplikasi pada sebuah mesin nyata. JVM menyediakan spesifikasi hardware dan platform dimana kompilasi kode Java terjadi. Spesifikasi inilah yang membuat aplikasi berbasis Java menjadi bebas dari *platform* manapun karena proses kompilasi diselesaikan oleh JVM.

  Aplikasi program Java diciptakan dengan *file* teks berekstensi *.java*. Program ini dikompilasi menghasilkan satu berkas *bytecode* berekstensi *.class* atau lebih. *Bytecode* adalah serangkaian instruksi serupa instruksi kode mesin. Perbedaannya adalah kode mesin harus dijalankan pada sistem komputer dimana kompilasi ditujukan, sementara *bytecode* berjalan pada *java interpreter* yang tersedia di semua *platform* sistem komputer dan sistem operasi.

- ✓ ***Garbage Collection***

  Banyak bahasa pemrogaman lain yang mengizinkan seorang *programmer* mengalokasikan memori pada saat dijalankan. Namun, setelah menggunakan alokasi memori tersebut, harus terdapat cara untuk menempatkan kembali blok memori tersebut supaya program lain dapat menggunakannya. Dalam C, C++ dan bahasa lainnya, adalah programmer yang mutlak bertanggung jawab akan hal ini. Hal ini dapat menyulitkan bilamana programmer tersebut alpa untuk mengembalikan blok memori sehingga menyebabkan situasi yang dikenal dengan nama *memory leaks.*

  Program Java melakukan *garbage collection* yang berarti program tidak perlu menghapus sendiri objek–objek yang tidak digunakan lagi. Fasilitas ini mengurangi beban pengelolaan memori oleh programmer dan mengurangi atau mengeliminasi sumber kesalahan terbesar yang terdapat pada bahasa yang memungkinkan alokasi dinamis.

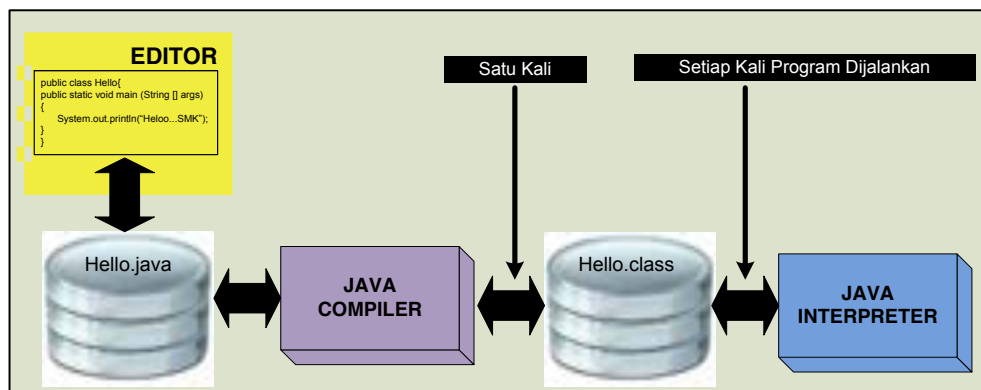
- ✓ ***Code Security***

  Code Security terimplementasi pada Java melalui penggunaan Java Runtime Environment (JRE). Java menggunakan model pengamanan 3 lapis untuk melindungi sistem dari *untrusted Java Code*.

- Pertama, *class-loader* menangani pemuatan kelas Java ke *runtime interpreter.* Proses ini menyediakan pengamanan dengan memisahkan kelas–kelas yang berasal dari *local disk* dengan kelas–kelas yang diambil dari jaringan. Hal ini membatasi aplikasi Trojan karena kelas–kelas yang berasal dari *local disk* yang dimuat terlebih dahulu.
- Kedua, *byte code verifie m*embaca *byte code* sebelum dijalankan dan menjamin *byte code* memenuhi aturan–aturan dasar bahasa Java.
- Ketiga, manajemen keamanan menangani keamanan tingkat aplikasi dengan mengendalikan apakah program berhak mengakses sumberdaya seperti system *file*, *port* jaringan, proses eksternal dan *system windowing*.

Setelah seluruh proses tersebut selesai dijalankan, barulah kode program di eksekusi.

✓ **Fase-Fase pemrograman Java**

Gambar di bawah ini menjelaskan aliran proses kompilasi dan eksekusi sebuah program Java

Gambar 9. Fase dari sebuah program Java

Langkah pertama dalam pembuatan sebuah program berbasis Java adalah menuliskan kode program pada *text editor*. Contoh *text editor* yang dapat digunakan antara lain: notepad, vi, emacs dan lain sebagainya. Kode program yang dibuat kemudian tersimpan dalam sebuah berkas berekstensi *.java.* Setelah membuat dan menyimpan kode program, kompilasi file yang berisi kode program tersebut dengan menggunakan Java Compiler. Hasil dari kompilasi berupa berkas *byte code* dengan ekstensi *.class.* Berkas yang mengandung *byte code* tersebut kemudian akan dikonversikan oleh Java Interpreter menjadi bahasa mesin sesuai dengan jenis dan *platform* yang

digunakan.

3) Instalasi Program Java SDK dan IDE Netbeans

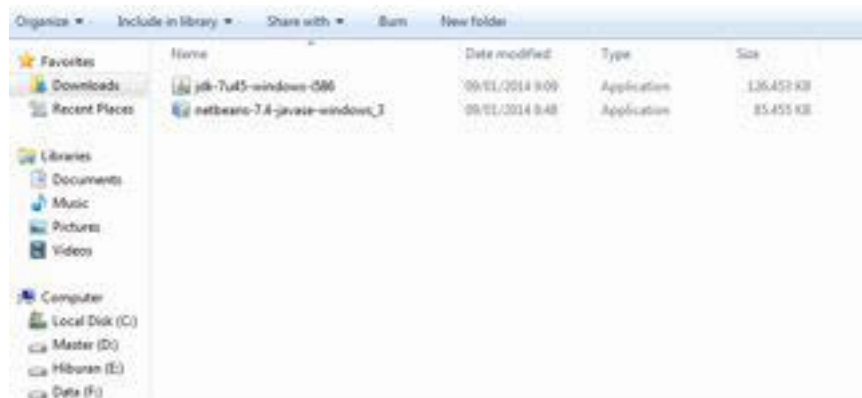
Java SDK dan NetBeans diperlukan jika anda hendak mulai membuat program dengan bahasa pemrograman Java. JavaSDK adalah platform dasar Java yang diperlukan agar komputer atau laptop dapat digunakan untuk mengeksekusi kode-kode programbahasa Java, sedangkan NetBeans adalah aplikasi editor terpadu (IDE atau Integrated Develepment Environment) yang akan banyak mempermudah dalam membuat aplikasi karena menyediakan kontrol-kontrol visual yang penting dalam pemrograman desktop (atau lebih dikenal sebagai pemrograman visual).

Sebelumnya, perlu diketahui bahwa JavaSDK dan NetBeans memerlukan sumberdaya yang cukup tinggi, jadi sama sekali tidak bisa dibandingkan dengan Delphi, C++ Builder, apalagi Visual Basic versi 6 atau sebelumnya. Disarankan mengunakan komputer atau laptop dengan minimal RAM 1 GB dan prosesor clock-speed diatas 1GHZ.

Berikut ini langkah-langkah instalasi Java SDK dan NetBeans pada PC dengan system operasi Windows XP Professional atau Windows Vista dan Windows 7 disertai screenshot.

✓ **Instalasi JavaSDK**

Java Standart Development Kit (SDK) tersedia untuk di download pada situs Web software Java Sun Microsystem pada :http://java.sun.com.

Open folder tempat file-file instalasi Java SDK. Dalam contoh ini file disimpan di drive D pada My Computer.

Gambar10. Open File Java Development Kit

Klik-dobel file instalasi Java SDK untuk mengeksekusi instalasi. Dalam contoh ini, file yang dieksekusi bernama lengkap jdk-7u45-windows-i586.exe. Dalam beberapa detik, akan muncul kotak dialog berisi lembar persetujuan antara pihak Sun Microsystems.Inc. Sebagai pembuat software dan pihak anda sebagai pengguna software

Gambar11.Destination Folder

Tahapan-tahapan proses instalasi Java SDK dapat dilakukan dengan mudah dengan mengikuti petunjuk proses instalasi dengan menekan button next sampai pada tahap finish. Saat instalasi selesai, muncul kotak dialog yang memberitakan bahwa instalasi Java SDK lewatkan dengan mengklik tombol *Finish*.

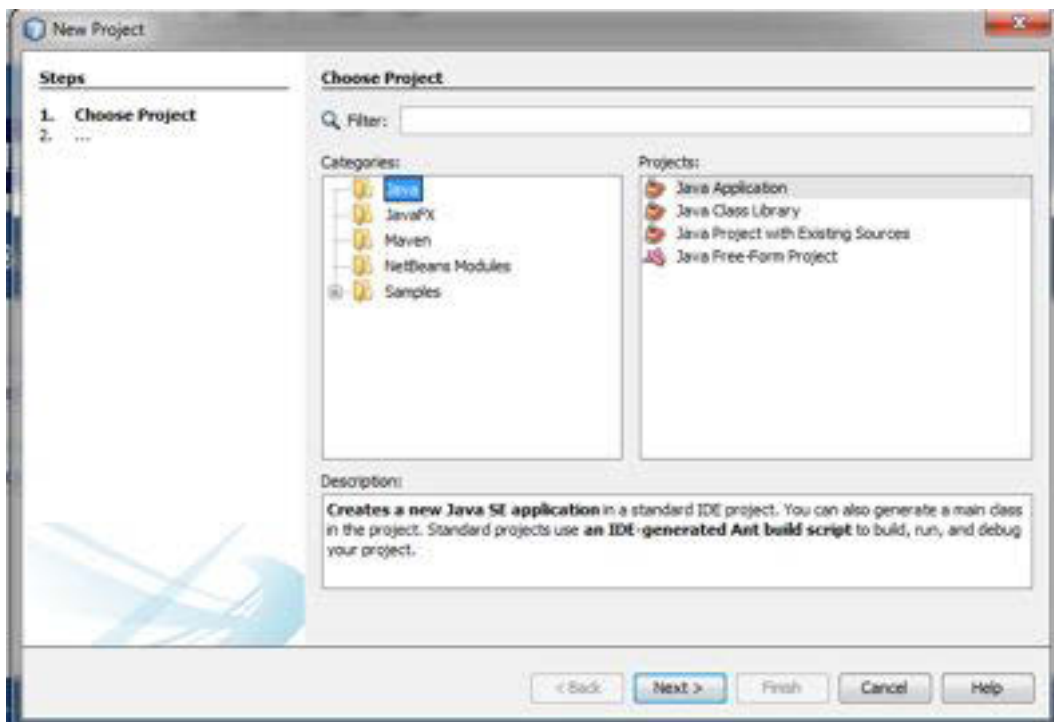
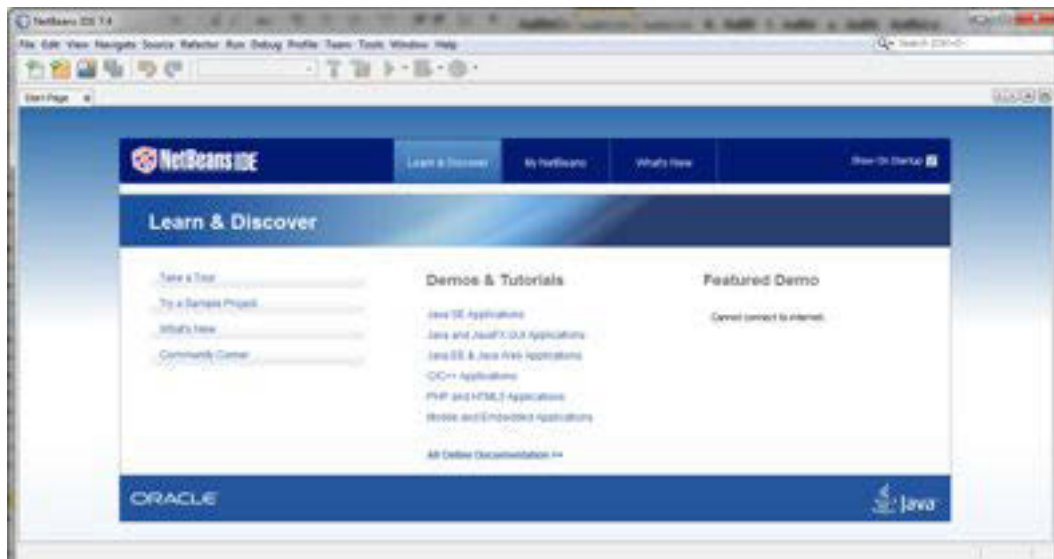
Gambar12.Proses Instalasi Berhasil

- ✓ **Instalasi NetBeans**

Instalasi Net Beans bisa dimulai dengan mengklik-dobel file instalasi yang ada di komputer atau laptop. Dalam contoh ini, filedi directory D, sehingga file **netbeans-7.4-javase-windows_3.exe.** Seperti halnya pada instalasi Java SDK atau produk-produk berlisensi lain, installer NetBeans akan menyodorkan halaman persetujuan antara pembuat software dan anda sebagai pemakai.

Gambar13.Licence Agreement NetBeans

Untuk memeriksa hasil instalasinya Netbens dengan langsung menggunakannya. Defaultnya, NetBeans bisa dibuka dengan mengklik ***Start*** pada Windows, pilih ***Programs***, pilih kelompok menu ***NetBeans***, lalu klik ***NetBeans IDE***.

Tanda NetBeans sedang dalam proses membuka modul-modul yang diperlukan untuk membuat aplikasi.

Gambar14.Start Up NetBeans

IDE NetBeans mengharuskan membuat *new Project* terlebih dahulu sebelum menulis *script* program java. Dengan cara klik File new Project , langkah berikutnya memilih aplikasi *Java Aplication*. File dengan *extension .java* dibuat untuk memulai menulis program java.

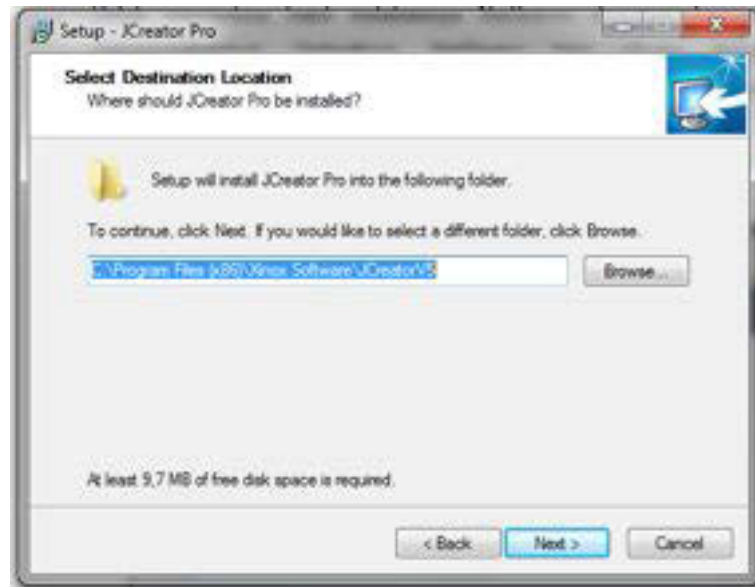
- ✓ **Instalasi JCreator 5 Pro**

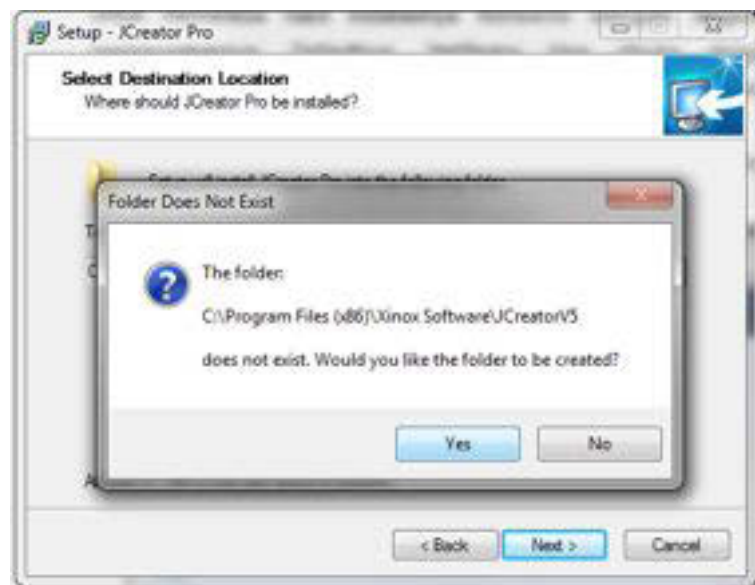
Instalasi JCreator Pro 5 diawali dengan membuka file setup Jcreator dengan cara mengklik-dobel yang ada di komputer anda. Selanjutnya jika anda sudah membuka file setup akan dimunculkan pernyataan seperti gambar di bawah:

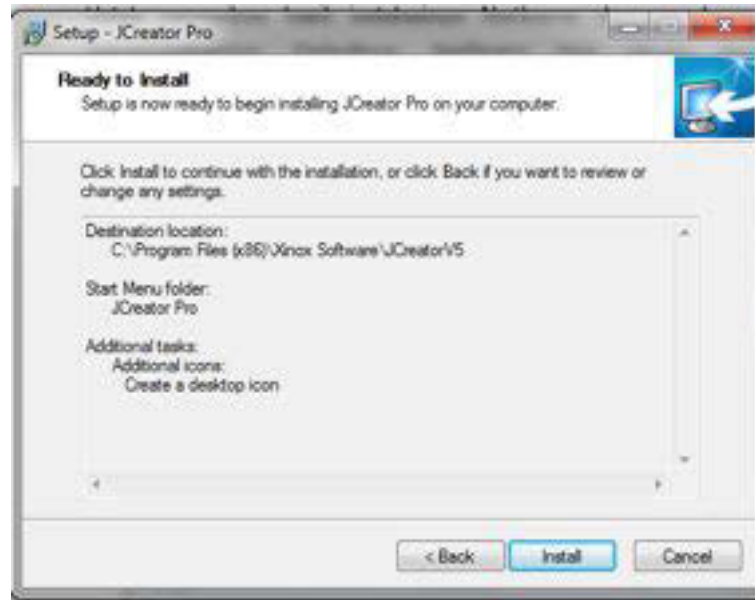
Gambar 15. Start Up Jcreator

Klik Next untuk melanjutkan proses instalasi, jika tidak ingin melanjutkan klik cancel.

Gambar 16. License Agreement JCreator

Klik lingkaran pada tulisan “*I accept the agreement*”.

Gambar 17. Destination Location Jcreator

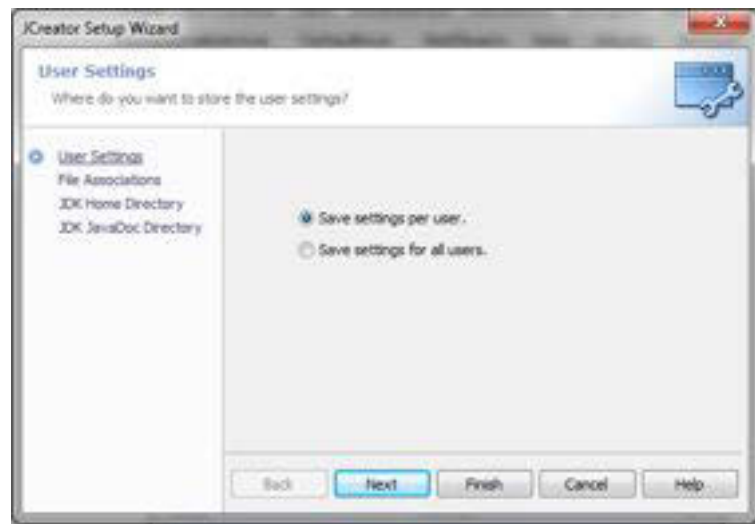
Klik next untuk melanjutkan instalasi pada drive C:\Program Files, jika anda ingin meletakkan di drive lain, anda bisa meng-klik tombol browse kemudian tentukan tempat yang anda inginkan.

Gambar 18. Create New Folder JCreator

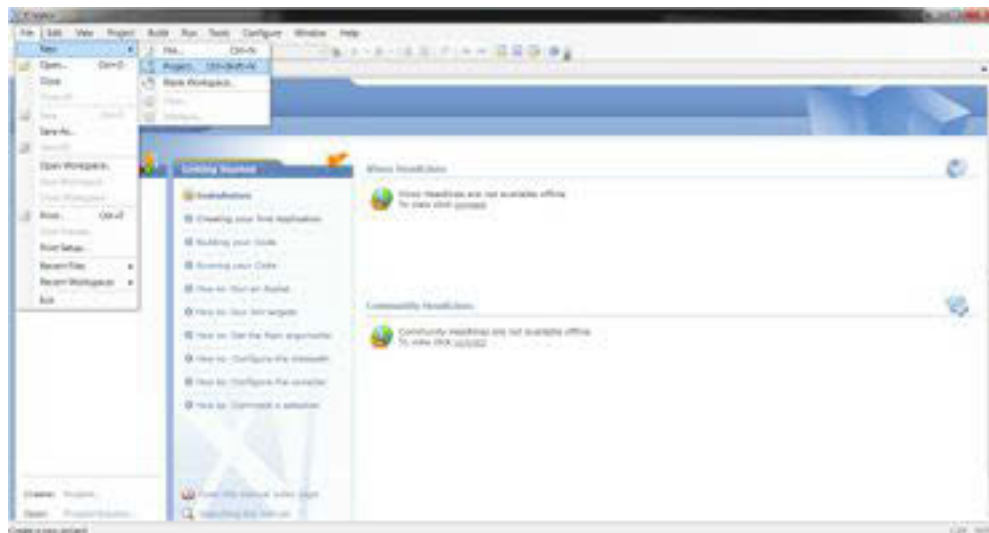
Klik “Yes” untuk membuat sebuah folder baru di C:\Program Files

Gambar 19. Install

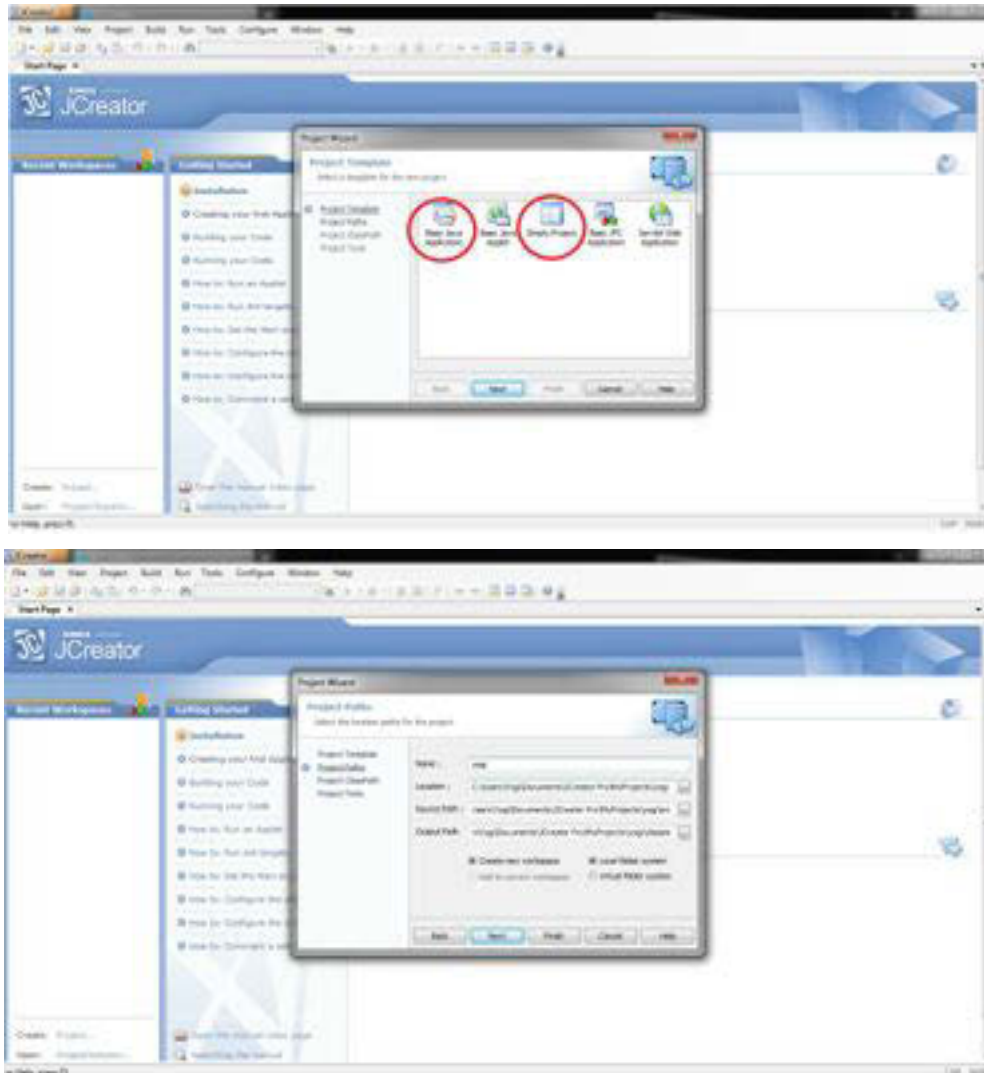
Klik Install dan tunggu proses hinga muncul tampilan seperti di bawah ini.

Gambar 20. Finishing Instalation

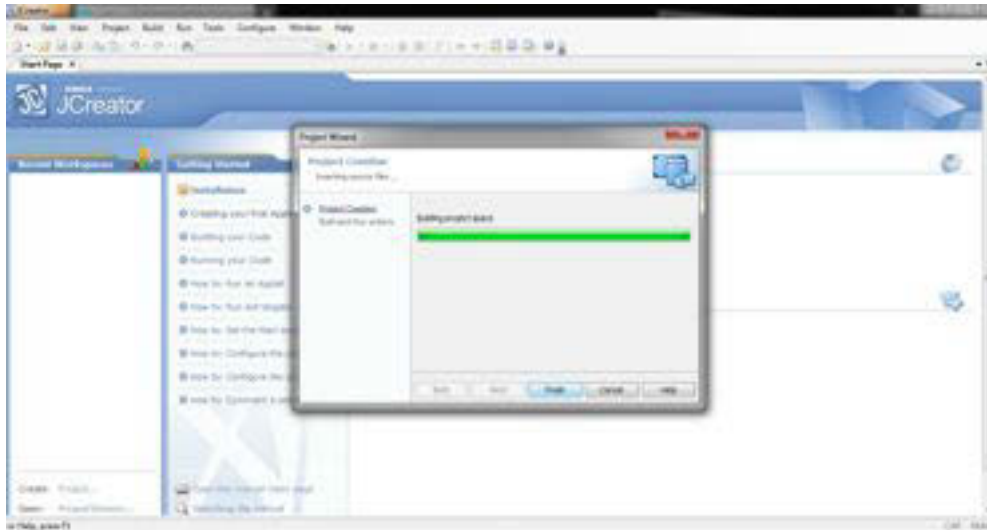
Klik “Finish” untuk menyelesaikan proses instalasi, Beri tanda centang tentang di bagian tulisan “Launch JCreator Pro” untuk membuka langsung program Jcreator dan jangan diberi jika tidak ingin membuka langsung.

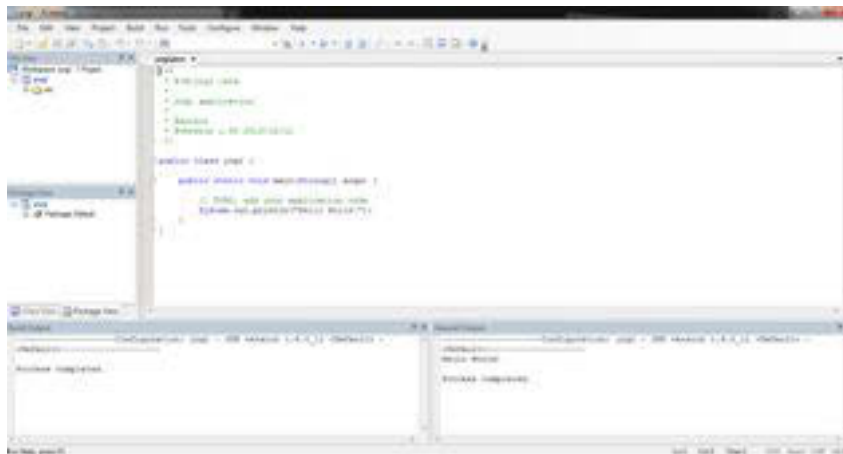
✓ **Membuat Project dengan Jcreator**

Gambar 21. JCreator Setup Wizard

Ketika pertama kali membuka JCreator maka 35ka nada tampilan seperti ini, klik Finish.

Gambar 22. Main Menu JCreator

Ini adalah tampilan menu utama pada JCreator Pro, klik File di bagian kiri atas, lalu new, dan kemudian klik Project.

Gambar 23. Project Wizard

Anda akan diminta untuk memilih Object Template, Jika anda pilih Basic Java Application maka akan memberikan project yang instan dengan program Hello World, Jika anda pilih Empty Project maka project belum terisi atau kosong.

Dan pada tampilan ini anda diminta untuk mengisikan sebuah nama project anda pada field name, kemudian klik next.

Gambar 24. Finishing Project

Tunggu proses sampai selesai kemudian klik finish.

Gambar 25. Jcreator

Dan inilah tampilan dari Basic Java Application yang berisi program Hello Word.

c. Rangkuman

Arsitektur teknologi Java dibagi menjadi tiga bagian yaitu (1) **Enterprise Java (J2EE)** untuk aplikasi berbasis web, aplikasi sistem tersebar dengan beraneka ragam klien dengan kompleksitas yang tinggi. Merupakan superset dari Standar Java (2) **Standar Java (J2SE),** ini adalah yang biasa dikenal sebagai bahasa Java. (3) **Micro Java (J2ME)** merupakan subset dari J2SE dan salah satu aplikasinya yang banyak dipakai adalah untuk wireless device / mobile device. Program Java dijalankan menggunakan interpreter yaitu *Java Virtual Machine* (JVM). Hal ini menyebabkan *source code* Java yang telah dikompilasi menjadi Java *byte codes* dapat dijalankan pada platform yang berbeda-beda.

Langkah pertama dalam pembuatan sebuah program berbasis Java adalah menuliskan kode program pada *text editor*. Contoh *text editor* yang dapat digunakan antara lain : notepad, vi, emacs dan lain sebagainya. Kode program yang dibuat kemudian tersimpan dalam sebuah berkas berekstensi *.java.* Setelah membuat dan menyimpan kode program,kompilasi file yang berisi kode program tersebut dengan menggunakan JavaCompiler. Hasil dari kompilasi berupa berkas *byte code* dengan ekstensi *.class.* Berkas yang mengandung *byte code* tersebut kemudian akan dikonversikan oleh Java Interpreter menjadi bahasa mesin sesuai dengan jenis dan *platform* yang digunakan.

d. Tugas

Tugas 1

Buatlah alur kerja sederhana dari perangkat lunak pemrograman berorientasi obyek.

❖ Mengamati benda dan obyek

1. Buatlah kelompok dengan anggota 3 – 4 orang
2. Amati langkah-langkah kerja untuk membuat sebuah kelas
3. Tuliskan langkah-langkah kerja sederhana perangkat berorientasi obyek
4. Deskripsikan setiap benda tersebut seperti gambar berikut

Nama Benda
Atribut :

Operasi :

6. Buat laporan dan diskusikan dengan teman sekelompok

❖ Bandingkan dan Simpulkan

Bandingkan alur kerja atau langkah-langkah dari hasil kerja kelompok anda dengan kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama

Tugas 2

Buatlah prosedur atau langkah-langkah menyajikan perangkat lunak pemrograman berorientasi obyek.

❖ Mengamati Bahasa Pemrograman

1. Buatlah kelompok dengan anggota 3 – 4 orang
2. Amatilah langkah-langkah menyajikan perangkat lunak pemrograman berorientasi obyek
5. Buatlah tabel perbedaan dan persamaan yang anda dapatkan diskusikan dengan teman sekelompok

No.	Langkah-langkah menyajikan perangkat lunak pemrograman berorientasi obyek
1.	
2.	

❖ Bandingkan dan Simpulkan

Bandingkan hasil tabel Langkah-langkah menyajikan perangkat lunak pemrograman berorientasi obyek dari hasil kerja kelompok anda dengan kelompok lain. Berdasarkan hasil pengamatam tersebut hal penting apa yang harus dirumuskan secara bersama ?

e. Test Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Arsitektur teknologi Java terbagi menjadi tiga, yaitu Enterprise Java (J2EE), Standar Java (J2SE), dan Micro Java (J2ME). Sebutkan device yang Anda temui yang menggunakan salah satu dari ketiga teknologi Java tersebut!
2. Sebutkan langkah-langkah yang harus Anda lakukan untuk menyajikan perangkat lunak pemrograman Java!
3. Sebutkan karakteristik Java menurut SUN!

f. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Device yang Anda temua yang menggunakan salah satu dari ketiga arsitektur teknoloi Java

a) Enterprise Java
(J2EE)..
....
....
....

b) Standar Java (J2SE)
…..
....
....
....

c) Micro Java
(J2ME)...
....
....
....

LJ- 02 : Langkah-langkah yang Anda lakukan untuk menyajikan perangkat lunak pemrograman berorientasi obyek.

....

g. Lembar Kerja Siswa

3. **Kegiatan Belajar 3 : Perangkat Lunak Pemrograman Berorientasi Obyek**

BAB II
ATURAN DAN DASAR PEMROGRAMAN BERORIENTASI OBYEK

A. Deskripsi

Dalam bab 2 ini akan menjelaskan dan menyajikan konsep pemrograman berorientasi obyek yang terdiri dari 4 kegiatan belajar. Kegiatan belajar 3 akan memahamkan anda tentang paradigma pemrograman berorientasi obyek dan menganalisis perbedaan pemrograman procedural dan pemrograman perorientasi obyek. Kegiatan belajar 4 meliputi penjelasan alur kerja perangkat lunak berorientasi obyek dan melakukan instalasi perangkat lunak. Setiap kegiatan belajar disertai dengan tujuan pembelajaran yang akan dicapai dalam 1 kali tatap muka, uraian materi, tes formatif untuk menguji kompetensi pengetahuan anda, dan tugas atau praktikum (individu dan kelompok) untuk menguji kompetensi keterampilan anda.

B. Kegiatan Belajar

1. Kegiatan Belajar 3: Dasar dan Aturan Pemrograman Berorientasi Obyek (Java Error, Keyword)

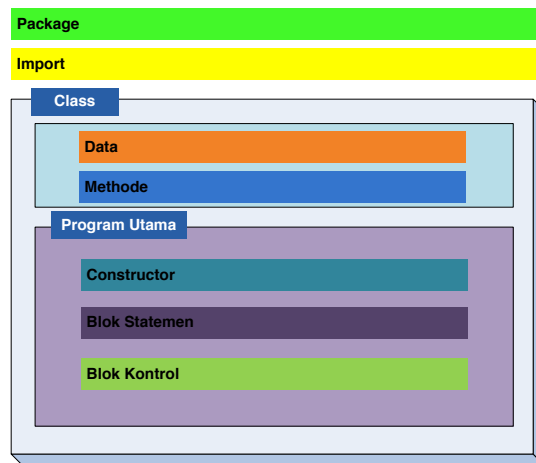
a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 3 ini siswa diharapkan dapat :

1) Mengidentifikasi bagian dasar dari program Java.
2) Memahami perbedaan antara syntax error dan runtime error.
3) Menganalisis Java literal, keyword, tipe data dasar, dan tipe variabel.

b. Uraian Materi

1) Bagan dasar program java

Gambar 26 .Bagian –bagian pemrograman Java

✓ **Package**

Perintah java yang digunakan untuk memberitahukan bahwa suatu class adalah anggota dari package, sedangkan nama Package dapat berupa susunan direktori tempat dimana file class disimpan atau nama folder.

✓ **Import**

Perintah import digunakan untuk memberitahukan kepada program untuk mengacu pada class-class yang terdapat pada package tersebut dan bukan menjalankan class-class tersebut**.** Dalam program, dapat diimport class-class tertentu saja dan dapat pula mengimport semua class yang terdapat pada package.

✓ **Class**

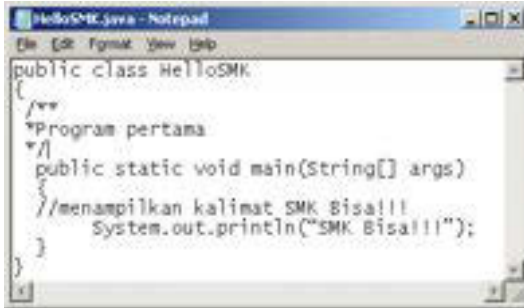
Merupakan bentuk logis yang menjadi landasan bangun seluruh bahasa pemograman berorientasi object. Class mendefinisikan bentuk dan perilaku object. Class merupakan contoh abstrak dari sebuah object yang telah terbentuk dari proses penyederhanaan. Dengan kata lain class merupakan cikal bakal dari object. Kemudian contoh nyata atau perwujudan dari sebuah object dinamakan instance.

✓ **Data dan Methode**

Data merupakan identitas yang berupa variabel yang menjelaskan properti dari class. Metoda adalah sekumpulan instruksi untuk menjalankan data yang diberi nama dan dapat dipanggil dari manapun di dalam program dengan menuliskan nama metoda tersebut.

✓ **Program utama**

Salah satu metoda yang paling penting di dalam bahasa Java adalah metoda

main. Metoda main harus dideklarasikan sendiri oleh programmer di dalam sebuah kelas. Kelas yang mempunyai metoda main disebut dengan kelas main (main class), akan tetapi tidak semua kelas Java harus mempunyai metoda main. Interpreter Java akan meminta metoda main saat program aplikasi dieksekusi.

Gambar 27. Bagian–bagian pemrograman Java

Baris pertama kode:

```
public class HelloSMK
```

menandakan nama class yaitu Hello. Dalam Java, semua kode seharusnya ditempatkan di dalam deklarasi class. Kita melakukannya dengan menggunakan kata kunci class.

Sebagai tambahan, class menggunakan *access specifier* **public**, yang mengindikasikan bahwa class kita mempunyai akses bebas ke class yang lain dari package yang lain pula (package merupakan kumpulan class-class). Baris berikutnya yaitu yang terdiri atas kurung kurawal {menandakan awal blok. Pada kode ini,kita menempatkan kurung kurawal pada baris selanjutnya setelah deklarasi class, bagaimanapun, kita dapat juga meletakkan kurung kurawal ini setelah baris pertama dari kode yang kita tulis. Jadi, kita dapat menulis kode kita sebagai berikut:

```
public class Hello
{
```

atau

```
public class Hello {
```

Tiga baris selanjutnya menandakan adanya komentar Java. Komentar adalah sesuatu yang digunakan untuk mendokumentasikan setiap bagian dari kode yang ditulis. Komentar bukan merupakan bagian dari program itu sendiri, tetapi digunakan untuk tujuan dokumentasi. Komentar itu sendiri dapat ditambahkan pada kode yang Anda tulis sebagai petunjuk yang dapat membantu proses pembelajaran pemrograman yang baik.

```
/**
* Program Pertama
*/
```

Komentar dinyatakan dengan tanda “/*” dan “*/”. Segala sesuatu yang ada

diantara tanda tersebut diabaikan oleh compiler Java, dan mereka hanya dianggap sebagai komentar. Baris selanjutnya,

```
public static void main(String[] args) {
```

atau dapat juga ditulis sebagai berikut,

```
public static void main(String[] args)
{
```

Mengindikasikan nama suatu method dalam class **Hello** yang bertindak sebagai **method utama.** Method utama adalah titik awal dari suatu program Java. Semua program kecuali applet yang ditulis dalam bahasa Java dimulai dengan method utama. Yakinkan untuk mengikuti kaidah penulisan tanda yang benar. Baris selanjutnya juga merupakan komentar,

```
//Menampilkan kalimat SMK Bisa!!!
```

Sekarang kita mempelajari 2 cara untuk membuat komentar. Cara pertama adalah dengan menempatkan komentar dalam /* dan */,dan cara yang lain adalah dengan menuliskan tanda//pada awal komentar. Baris selanjutnya,

```
System.out.println("SMK Bisa!!!");
```

menampilkan teks “HelloWorld!” pada layar. Perintah System.out.println(), menampilkan teks yang diapit oleh tanda *doublepute* (“”) pada layar. Dua baris terakhir yang terdiri atas dua kurung kurawal digunakan untuk menutup method utama dan masing-masing class secara berurutan.

1. Program Java yang Anda buat harus selalu diakhiri dengan ekstensi file .java.
2. Nama File seharusnya sesuai/sama dengan nama class public nya. Sebagai contoh, jika nama class public Anda adalah HelloSMK, Anda harus menyimpan file tersebut dengan nama HelloSMK.java.
3. Anda harus menulis komentar sebagai penjelasan pada kode yang Anda tulis, yaitu komentar yang berisi keterangan mengenai baris perintah pada class atau apa yang dijalankan oleh method yang Anda tulis tersebut.

2) Perbedaan Syntax Error dan Runtime Error

✓ **Syntax Error**

Syntax Error biasanya terjadi karena kesalahan penulisan. Mungkin kekurangan sebuah perintah di Java atau lupa untuk menulis tanda titik koma pada akhir pernyataan. Java mencoba untuk megisolasi error tersebut dengan cara menunjukkan baris dari kode dan terlebih dahulu karakter yang salah dalam baris tersebut. Bagaimanapun juga, error belum tentu berada

pada titik yang ditunjuk.

Kesalahan umum lainnya adalah dalam kapitalisasi, ejaan, penggunaan dari karakter khusus yang tidak benar, dan penghilangan dari pemberian tanda baca yang sebenarnya. Mari kita mengambil contoh pada program HelloSMK.java, dimana dengan sengaja kita menghilangkan titik koma pada akhir pernyataan dan juga mencoba untuk mengetikkan ejaan yang salah pada sebuah perintah.

Lihatlah pesan error yang ditampilkan setelah peng-compile-an program dilanjutkan. Pesan error yang pertama memberitahu bahwa di program dijalankan. Pada error yang pertama memberitahu bahwa di program terdapat error pada baris 6. Hal itu menunjuk pada kata setelah static, dimana seharusnya dieja sebagai static. Pada error yang kedua memberitahukan bahwa pada program tersebut kehilangan titik-koma setelah pernyataan.

✓ **Runtime Error**

Sebuah program yang berhasil dikompile belum tentu berhasil dijalankan. Inilah yang dinamakan Run time error, kesalahan ini tidak akan ditampilkan sampai kita menjalankan program tersebut. Hal ini bisa saja terjadi misalnya dikarenakan struktur yang dibuat programmer tidak jelas atau mungkin tidak logis.

3) Java literal, keyword, tipe data dasar, dan tipe variabel

✓ **Java Keywords**

Kata-kunci (keywords) dari sebuah bahasa pemrograman adalah kata-kata yang didefinisikan secara khusus yang hanya dimengerti oleh compiler bahasa pemrograman tersebut, dan tidak dapat digunakan sebagai identitas variabel. Di bawah ini ditampilkan semua kata-kunci (Java keywords) :

Tabel 1. Java Keywords

Abstract	Default	If	private	this
Boolean	Do	implements	protected	throw
Break	Double	import	public	throws
Byte	Else	instanceof	return	transient
Case	Extends	int	short	try
Catch	Final	interface	static	void
Char	Finaly	long	strictfp	volatile

Class	Float	native	super	while
Const	For	new	switch	
Continue	Goto	package	synchronized	assert

Catatan: true, false, dan null bukan termasuk kata kunci akan tetapi mereka termasuk kata-kata khusus, jadi Anda tidak dapat menggunakan mereka sebagai nama variabel pada program Anda.

✓ **Java Literals**

Literals adalah tanda bahwa tidak terjadi perubahan atau konstan. Macam-macam literals dalam Java adalah : *Integer Literals*, *Floating-Point Literals*, *Boolean Literals*, *Character Literals* dan *String Literals*.

1. ***Integer Literals***

   Integer literals dibedakan dalam beberapa format yang berbeda: desimal (berbasis 10), heksadesimal (berbasis 16), and oktal (berbasis 8). Dalam penggunaan tipe data integer pada program, kita harus mengikuti aturan penggunaan beberapa notasi khusus.Untuk angka desimal, kita tidak memerlukan notasi khusus. Kita hanya menulis angka desimal seperti apa adanya. untuk angka heksadesimal, hal itu harus ditandai oleh "0x" atau "0X". untuk oktal, ditandai oleh"0". Sebagai contoh, mewakili angka **12**. Penulisan dalam bentuk desimalnya adalah **12**, Sementara dalam heksadesimal, menjadi **0xC**, dan dalam oktal, nilai tersebut sama dengan **014**.

   Default tipe data untuk integer literals adalah **int**. Int adalah signed 32-bit value. Pada kasus-kasus tertentu Anda dapat berharap untuk memaksa integer literal untuk menjadi tipe data **long** dengan menambahkan karakter "l" or "L". tipe data long ditandai oleh ditampilkannya data dalam 64-bit.

2. ***Floating-Point Literals***

   Floating point literals mewakili bentuk desimal dengan bagian yang terpisah. Sebagai contoh adalah 3.1415. *Floating point literals* dapat dinyatakan dalam notasi standard atau scientific. Sebagai contoh, 583.45 dinyatakan dalam notasi standard, Sementara 5.8345e2 dinyatakan dalam notasi scientific. Default *Floating point literals* mempunyai tipe data **double** yang dinyatakan dalam 64-bit. Untuk menggunakan ketelitian yang lebih kecil (32-bit) **float**, hanya dengan menambahkan karakter "f" atau "F".

3. ***Boolean Literals***

   Boolean literals hanya memiliki dua nilai, true atau false.

4. ***Character Literals***

   Character Literals diwakili oleh karakter single Unicode. Karakter Unicode adalah 16-bit character set yang menggantikan 8-bit ASCII character set. Unicode memungkinkan pengunaan simbol dan karakter khusus dari bahasa lain. Untuk menggunakan *character literals*, karakter tersebut di dalam tanda *single pute* (' ') (single quote delimiters). Sebagai contoh huruf a, diwakili sebagai **'a'**. Untuk menggunakan karakter khusus seperti karakter baris baru, *backslash* digunakan diikuti dengan karakter kode. Sebagai contoh, '\n' untuk karakter baris baru atau ganti baris, '\r' untuk menyatakan nilai balik (carriage return), '\b' untuk backspace.

5. ***String Literals***

   String literals mewakili beberapa karakter dan dinyatakan dalam tanda double pute (" ")(double quotes). Sebagai contoh *string literal* adalah, **"Hello World".**

E. Tipe Data Primitif

Bahasa pemrograman Java mendefinisikan delapan tipe data primitif. Mereka diantaranya adalah boolean (untuk bentuk logika), char (untuk bentuk tekstual), byte, short, int, long (integral), double and float (floating point).

1. ***logika - boolean***

   Tipe data boolean diwakili oleh dua pernyataan : true dan false. Sebagai contoh adalah, boolean result = true; Contoh yang ditunjukkan diatas, mendeklarasikan variabel yang dinamai **result** sebagai tipe data **boolean** dan memberinya nilai **true**.

2. ***teksual – char***

   Tipe data character (char), diwakili oleh karakter single Unicode. Tipe data ini harus memiliki ciri berada dalam tanda *single quotes*(' '). Sebagai contoh,

```
 'a'    //Huruf a
'\t'   //A tab
Untuk menampilkan karakter khusus seperti ' (single quotes) atau " (double quotes), menggunakan karakter
```

```
escape \. Sebagai contoh,
'\''  //untuk single quotes
'\"'  //untuk double quotes
```

Meskipun String bukan merupakan tipe data primitif (namun merupakan suatu Class), kita akan memperkenalkan mengenai pada bagian ini. String mewakili tipe data yang terdiri atas beberapa karakter. Mereka tidak **termasuk tipe data primitif, melainkan suatu class.** Mereka memiliki literal yang terdapat diantara tanda double quotes(“”).

Sebagai contoh,

```
String message="Hello world!"
```

3. ***Integral –byte, short, int & long***

   Tipe data integral dalam Java menggunakan tiga bentuk- yaitu desimal,oktal atau heksa desimal. Contohnya,

   ```
   2    //nilai desimal 2
   077  //angka 0awal mengindikasikan nilai oktal
   0xBACC //karakter 0x mengindikasikan nilai
   heksadesimal
   ```

   Tipe-tipe integral memiliki default tipe data yaitu int. Anda dapat merubahnya ke bentuk long dengan menambahkan huruf l atau L

4. ***Floating Point –float dan Double***

   Tipe Floating point memiliki double sebagai default tipe datanya. Floating-point literal termasuk salah satunya decimal point atau salah satu dari pilihan berikut ini:

   E or e //(add exponential value) F or f //(float)
   D or d //(double)
   Contohnya adalah,

   3.14 //nilai floating-point sederhana (a double)
   6.02E23 //A nilai floating-point yang besar
   2.718F //A nilai float size sederhana
   123.4E+306D //A nilai double yang besar dengan nilai redundant D

   Pada contoh yang ditunjukkan diatas, 23 setelah E pada contoh kedua bernilai positif. Contoh tersebut sama dengan 6.02E+23. Java adalah bahasa pemrograman yang bertipe kuat. Ini maksudnya adalah setiap variabel harus memiliki sebuah tipe yang telah dideklarasikan dan bahasa tersebut memberlakukan pemerikasaan tipe yang kaku.

Bahasa Java mempunyai delapan tipe primitif yang ditunjukkan dalam tabel di bawah ini:

Tabel 2. Tipe Data Primitif

Grup	Type Data	Size	Min Value	Max Value
Integral	byte	8 bits	-128	128
	short	16 bits	-32768	32768
	int	32 bits	-2147483648	2147483648
	long	64 bits	-9223372036854775808	9223372036854775808
Real	float	32 bits	± 1.40239846E-45	±3.40282347E+8
	double	64 bits	±4.94065645841246544E-324	±1.79769313486231570E+308
Karakter	char	16 bits	\u0000	\uFFFF
Boolean	boolean	n/a	true atau false	

G. Variabel

Variabel adalah item yang digunakan data untuk menyimpan pernyataan objek. Variabel memiliki tipe data dan nama. Tipe data menandakan tipe nilai yang dapat dibentuk oleh variabel itu sendiri. Nama variabel harus mengikuti aturan untuk identifier.

a. Deklarasi dan Inisialisasi Variabel

Untuk deklarasi variabel adalah sebagai berikut,

```
<data tipe><name> [=initial value];
```

Catatan: Nilainya berada diantara <> adalah nilai yang disyaratkan, sementara nilai dalam tanda [] bersifat optional. Berikut ini adalah contoh program yang mendeklarasikan dan menginisialisasi beberapa variabel,

Listing Program

```
short x;
int umur;
float gaji;
double data;
```

Inisialisasi variabel dapat dilakukan dengan memberikan nilai pada variabel yang telah dideklarasikan, contoh :

Listing Program

```
int x=21;
int y;
double d = 3.5;
y = (int) d;
```

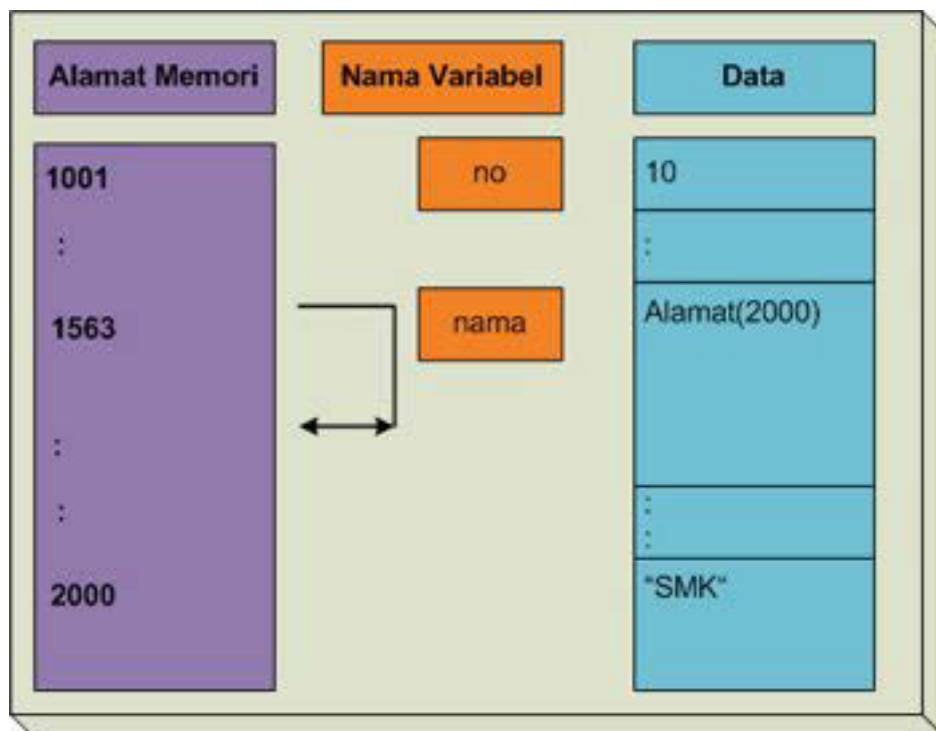
b. ***Variabel Reference dan Variabel Primitif***

Sekarang kita akan membedakan dua tipe variabel yang dimiliki oleh program Java. Ada **variabel reference** dan **variabel primitif** .**Variabel primitif** adalah variabel dengan tipe data primitif. Mereka menyimpan data dalam lokasi memori yang sebenarnya dimana variabel tersebut berada.

Variabel Reference adalah variabel yang menyimpan alamat dalam lokasi memori. Yang menunjuk ke lokasi memori dimana data sebenarnya berada. Ketika Anda mendeklarasikan variabel pada class tertentu, Anda sebenarnya mendeklarasikan reference variable dalam bentuk objek dalam classnya tersebut. Sebagai contoh, apabila kita mempunyai dua variabel dengan tipe data int dan String.

Listing Program

```
int no = 10;
String nama = "SMK ";
```

Dimisalkan ilustrasi yang ditunjukkan di bawah ini adalah memori yang ada pada komputer Anda, dimana Anda memiliki alamat dari setiap sel memorinya, nama variabel dan datanya terbentuk sebagai berikut.

Gambar 28. Ilustrasi Memori pada Komputer

c. Rangkuman

Pengembangan berorientasi objek merupakan cara pikir baru tentang perangkat lunak berdasarkan abstraksi yang terdapat dalam dunia nyata. Dalam konteks pengembangan menunjuk pada bagian awal dari siklus hidup pengembangan sistem, yaitu survei, analisis, desain, implementasi, dan pemeliharaan sistem. Hal yang lebih penting dalam pengembangan berorientasi objek adalah konsep mengindentifikasi dan mengorganisasi domain aplikasi dibandingkan dengan fokus penggunaan bahasa pemrograman, berorientasi objek atau tidak.

Object adalah gabungan antara beberapa data dan fungsi yang masing-masing bekerja bersama-sama dan tidak dapat dipisahkan. Gabungan dari data dan fungsi tersebut akan membentuk suatu object-object yang aktif. Dari kumpulan beberapa object yang sama akan membentuk struktur baru yang disebut class.
Pemrograman berorientasi objek (Inggris: object-oriented programming disingkat OOP) merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya.
Pemrograman terstruktur adalah suatu proses untuk mengimplementasikan urutan langkah untuk menyelesaikan suatu masalah dalam bentuk program. Selain pengertian diatas Pemrograman Terstruktur adalah suatu aktifitas pemrograman dengan memperhatikan urutan langkah-langkah perintah secara sistematis, logis, dan tersusun berdasarkan algoritma yang sederhana dan mudah dipahami. Prinsip dari pemrograman terstruktur adalah Jika suatu proses telah sampai pada suatu titik / langkah tertentu , maka proses selanjutnya tidak boleh mengeksekusi langkah sebelumnya / kembali lagi ke baris sebelumnya, kecuali pada langkah – langkah untuk proses berulang (Loop).

d. Tugas

Tugas 1

Buatlah listing program untuk kelas Meja dengan menampilkan karakteristik dari kelas tersebut, serta tunjukkan bagian-bagian listing programnya.

❖ Mengamati benda dan obyek

1. Buatlah kelompok dengan anggota 3 – 4 orang
2. Amatilah obyek di lingkungan sekitar Anda
3. Sebutkan ciri-ciri atau atribut dari obyek tersebut sehingga mudah dikenali.
4. Sebutkan fungsi dari obyek tersebut yang merupakan operasi relasi dari ciri-ciri yang sudah teridentifikasi
5. Deskripsikan setiap benda tersebut seperti gambar berikut

Nama Benda
Atribut :
Operasi :

6. Buat laporan dan diskusikan dengan teman sekelompok

❖ Bandingkan dan Simpulkan

Bandingkan urutan listing program telah anda buat dengan listing program teman anda. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

Tugas 2

Buatlah listing program untuk kelas Siswa yang menampilkan method dari Mobil tersebut. Tunjukkan Java literal, keyword, tipe data dasar, dan tipe variabel di dalamnya.

❖ Mengamati Bahasa Pemrograman

1. Buatlah kelompok dengan anggota 3 – 4 orang
2. Amatilah obyek (siswa) di sekitar Anda
3. Sebutkan vava literal, keyword, tipe data dasar, dan tipe variabel di dalamnya dari obyek tersebut
5. Buat laporan dan diskusikan dengan teman sekelompok

❖ Bandingkan dan Simpulkan

Bandingkan urutan listing program yang telah anda buat dengan listing program

teman anda. Berdasarkan hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama?

e. Test Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Coba amati sebuah obyek di lingkungan sekitar Anda. Selanjutnya tentukan package, import, class, data, dan method-nya.
2. Sebutkan Java literal, keyword, tipe data dasar, dan tipe variabel dari sebuah obyek yang Anda amati.

f. Lembar Jawaban Test Formatif (LJ)

LJ- 01 : Ilustrasi dalam dunia nyata :

a) Package...
....
....
....

b) Import ....
....
....
....

c) Methode...
....
....
....

d) Kelas…..
....
....
....

2. Kegiatan Belajar 4 :Dasar dan Aturan Pemrograman Berorientasi Obyek (Operator Logika)

a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 4 ini siswa diharapkan dapat :

1) Mengidentifikasi operator dalam program Java.
2) Menyajikan dalam perbedaan antara syntax error dan runtime error.

b. Uraian Materi

1) Operator

Dalam Java, ada beberapa tipe operator. Ada operator aritmatika, operator relasi, operator logika, dan operator kondisi. Operator ini mengikuti bermacam-macam prioritas yang pasti sehingga compilernya akan tahu yang mana operatoruntuk dijalankan lebih dulu dalam kasus beberapa operator yang dipakai bersama-sama dalam satu pernyataan.

3. *Operator Aritmatika*

Berikut ini adalah dasar operator aritmatik yang dapat digunakan untuk membuat suatu program Java,

Tabel 3. Operator Aritmatika dan Fungsi-Fungsinya

Operator	*Penggunaan*	*Keterangan*
+	op1+ op2	Menambahkan op1 dengan op2
*	op1*op2	Mengalikan op1 dengan op2
/	op1/op2	Membagi op1 dengan op2
%	op1%op2	Menghitung sisa dari pembagian op1 denganop2
-	op1-op2	Mengurangkan op2 dari op1

4. *Operator Increment dan Decrement*

Dari sisi operator dasar aritmatika, Java juga terdiri atas operator *unary increment* (++) dan operator *unary decrement* (--).Operator increment dan decrement menambah dan mengurangi nilai yang tersimpan dalam bentuk variabel angka terhadap nilai 1. Sebagai contoh,

```
pernyataan,
count = count + 1
count++;
```

Tabel 4. Operator Increment dan Decrement

Operator	*Penggunaan*	*Keterangan*
++	op++	Menambahkan nilai 1 pada op; mengevaluasi nilai op sebelum diincrementasi/ ditambahkan
++	++op	Menambahkan nilai 1 pada op; mengevaluasi nilai op setelah diincrementasi/ ditambahkan
--	op--	Mengurangkan nilai 1 pada op; mengevaluasi nilai op sebelum didecrementasi/ dikurangkan
--	--op	Mengurangkan nilai 1 pada op; mengevaluasi nilai op setelah didecrementasi/ dikurangkan

Operator increment dan decrement dapat ditempatkan sebelum atau sesudah operand. Ketika digunakan sebelum operand, akan menyebabkan variabel diincrement atau didecrement dengan nilai1, dan kemudian nilai baru digunakan dalam pernyataan dimana dia ditambahkan. Sebagai contoh,

Listing Program

```
int i =10;
int j = 3;
int k = 0;
k = ++j + i;
```

Ketika operator increment dan decrement ditempatkan setelah operand, nilai variabel yang lama akan digunakan lebih dulu dioperasikan lebih dulu terhadap pernyataan dimana dia ditambahkan. Sebagai contoh,

Listing Program

```
int i = 10,
    int j = 3;
    int k = 0;
    k = j++ + i;
```

5. ***Operator Relasi***

Operator Relasi membandingkan dua nilai dan menentukan keterhubungan diantara nilai- nilai tersebut. Hasil keluarannya berupa **nilai boolean** yaitu true atau false.

Tabel 5. Operator Relasi

Operator	***Penggunaan***	***Keterangan***
>	op1>op2	op1 lebih besar dari op2
>=	op1>= op2	op1 lebih besar dari atau sama dengan op2
<	op1<op2	op1 kurang dari op2
<=	op1<= op2	op1kurang dari atau sama dengan op2
==	op1== op2	op1 sama dengan op2
!=	op1!= op2	op1 tidak sama dengan op2

6. ***Operator logika***

Operator logika memiliki satu atau lebih operand Boolean yang menghasilkan nilai boolean. Terdapat enam operator logika yaitu : && (logika AND), & (Boolean logika AND), || (logika OR), | (Boolean logika inclusive OR), ^ (Boolean logika exclusive OR), dan ! (logika NOT).

Pernyataan dasar untuk operasi logika adalah x1 op x2, dimana x1,x2 dapat menjadi pernyataan boolean. Variabel atau konstanta, dan op adalah salah satu dari operator &&, &, ||, | atau ^. Tabel kebenaran yang akan ditunjukkan selanjutnya, merupakan kesimpulan dari hasil dari setiap operasi untuk semua kombinasi yang mungkin dari x1 dan x2.

4.1 (logika AND) dan & (Boolean logika AND)

Berikut ini adalah tabel kebenaran untuk && dan &.

Tabel 6. Tabel Kebenaran Logika AND

x1	***x2***	***Hasil***
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

Perbedaan dasar antara operator && dan & adalah bahwa && mensupports **short-circuit evaluations** (atau evaluasi perbagian), sementara operator & tidak. Apa arti dari pernyataan tersebut?

Diberikan suatu pernyataan,

```
exp1 && exp2
```

&& akan mengevaluasi pernyataan exp1, dan segera mengembalikan nilai false dan menyatakan bahwa exp1 bernilai false. Jika exp1 bernilai false, operator tidak akan pernah mengevaluasi exp2 karena hasil operasi operator akan menjadi false tanpa memperhatikan nilai dari exp2. Sebaliknya, operator & selalu mengevaluasi ke dua nilai dari exp1 dan exp2 sebelum mengembalikan suatu nilai jawaban.

4.2. II (logikaOR) danI(Boolean logika inclusive OR)

Berikut ini adalah tabel kebenaran untuk || dan |.

Tabel 7. Tabel Kebenaran Logika OR

x1	***x2***	***Hasil***
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

Perbedaan dasarantara operator|| dan | adalah bahwa || mendukung short-circuit evaluations (atau proses evaluasi sebagian), sementara | tidak. Apa maksud dari pernyataan tersebut? Diberikan suatu pernyataan,

```
exp1 || exp2
```

|| akan mengevaluasi pernyataan exp1,dan segera mengembalikan nilai true dan menyatakan bahwa exp1 bernilaitrue. Jika exp1 bernilai true, operator tidak akan pernah mengevaluasi exp2 karena hasil dari operasi operatorakan bernilai true tanpa memperhatikan nilai dari exp2. Sebaliknya, operator | selalu mengevaluasi ke dua nilai dari exp1 and exp2 sebelum mengembalikan suatu jawaban suatu nilai.

4.3. ^ (Boolean logika Exclusive OR)

Berikut ini adalah tabel kebenaran untuk^.

Tabel 8. Tabel Kebenaran Logika EX-OR

x1	*x2*	*Hasil*
TRUE	TRUE	FALSE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

Hasil operasi operator exclusive OR adalah TRUE, jika dan hanya jika satu operand bernilai TRUE dan yang lain bernilai False. Catatan jika kedua operand harus selalu dievaluasi untuk menjumlahkan hasil dari suatu exclusive OR.

4.4 ! (logika NOT)

Logika NOT digunakan dalam satu argumen, dimana argument tersebut dapat menjadi suatu pernyataan, variable atau konstanta. Berikut ini adalah tabel kebenaran untuk operator not !

Tabel 9. Tabel Kebenaran Logika NOT

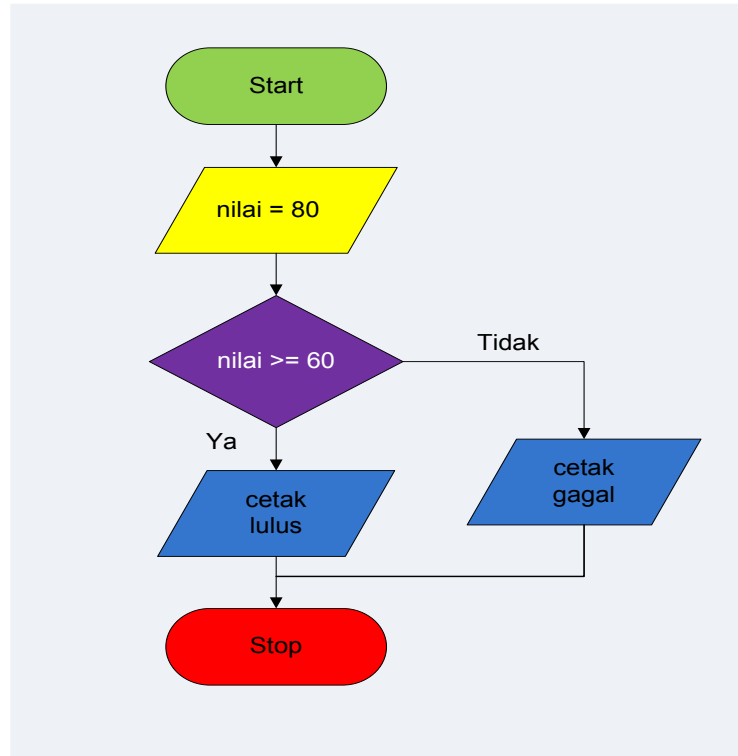
x1	*Hasil*
TRUE	FALSE
FALSE	TRUE

1. ***Operator Kondisi (?:)***

Operator kondisi **?:** adalah operator ternary. Berarti bahwa operator ini membawa tiga argumen yang membentuk suatu ekspresi bersyarat. Struktur pernyataan yang menggunakan operator kondisi adalah,

```
exp1?exp2:exp3
```

Dimana nilai exp1 adalah suatu pernyataan Boolean yang memiliki hasil yang salah satunya harus berupa nilai true atau false.

Jika exp1 bernilai true, exp2 merupakan hasil operasi. Jika bernilai false, kemudian exp3 merupakan hasil operasinya. Berikut ini adalah flowchart yang menggambarkan bagaimana operator ?: bekerja,

Gambar 29. Flowchart Operator Kondisi

2. ***Operator Precedence***

Operator precedence didefinisikan sebagai perintah yang dilakukan compiler ketika melakukan evaluasi terhadap operator, untuk mengajukan perintah dengan hasil yang tidak ambigu/hasil yang jelas.

```
 .     [ ]    ( )
 ++    --     !      ~
 *     /      %
 +     -
 <<    >>     >>>    <<<
 <     >      <=     >=
 ==    !=
 &     |
 ^
 &&
 ||
 ?:
 =
```

Gambar 30. Operator Presedence

Diberikan pernyataan yang membingungkan,

```
6%2*5+4/2+88-10
```

Kita dapat menuliskan kembali pernyataan diatas dan menambahkan beberapa tanda kurung terhadap operator precedence,

```
((6%2)*5)+(4/2)+88-10;
```

c. Rangkuman

Ada operator aritmatika, operator relasi, operator logika, dan operator kondisi. Operator ini mengikuti bermacam-macam prioritas, operator aritmatika umumnya digunakan untuk operasi matematika seperti pembagian, perkalian, dan lain-lain. Operator Increment dan Decrement berfungsi untuk menambah dan mengurangi nilai yang tersimpan, operasi relasi digunakan untuk membandingkan dua nilai untuk menentukan keterhubungan diantara nilai-nilai tersebut. Operator logika memiliki lebih dari satu operand boolean yang menghasilkan nilai boolean true dan false. Operator kondisi bersifat ternary, jadi operator ini membawa tiga argumen yang membentuk suatu ekspresi bersyarat. Operator Precedence berfungsi sebagai alat evaluasi terhadap operator untuk mengajukan perintah dengan hasil yang tidak ambigu atau hasil yang jelas.

d. Tugas

Tugas 1

Tulislah suatu program yang bisa menentukan nilai paling besar dari tiga bilangan *integer.* Ketiga integer disimpan dalam variabel **angka1** , **angka2** , dan **angka3**. Program bisa menentukan bilangan yang paling besar dari ketiga angka tersebut.

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Gambar Class Diagram

Nama Class
Atribut :
Operasi :

4. Buatlah listing program
5. Compile dan debug program

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

Tugas 2

Buatlah program untuk menghitung suatu harga barang yang bernilai Rp. 200.000 dengan diskon 15%.

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Gambar Class Diagram

4. Buatlah listing program
5. Compile dan debug program

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

e. Test Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Jelaskan secara singkat apa yang disebut :
   a. Operasi Aritmatika
   b. Operator Increment dan Decrement
   c. Operator Relasi
   d. Operator Logika
2. Sebutkan minimal 5 operator pada java yang anda ketahui
3. Sebutkan enam operator logika yang kamu ketahui
4. Operator kondisi adalah ternary, apakah yang dimaksud dengan ternary dan beri contohnya.

f. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Jelaskan secara singkat apa yang disebut:

a) Operator Aritmatika

....

....

....

....

b) Operator Increment dan Decrement

…..

....

....

....

c) Operator Logika

....

....

....

....

d) Kelas

……..

....

LJ- 02 : Sebutkan minimal 5 operator pada java yang anda ketahui !

....

....

....

....

....

....

....

LJ- 03 : Sebutkan enam operator logika yang kamu ketahui !

....

....

....

....

....

....

....

....

LJ- 04: Operator kondisi adalah ternary, apakah yang dimaksud dengan ternary dan beri contohnya. !

....

....

....

....

....

....

....

....

....

g. Lembar Kerja Siswa

3. Kegiatan Belajar 5 :Dasar dan Aturan Pemrograman Berorientasi Obyek (Kondisi)

a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 5 ini siswa diharapkan dapat :

1) Memahami struktur kontrol pemilihan (if, else, switch)
2) Menggunakan struktur kontrol pemilihan (if, else, switch) yang digunakan untuk memilih blok kode yang akan dieksekusi

b. Uraian Materi

Struktur kontrol pemilihan adalah pernyataan dari Java yang mengijinkan user untuk memilih dan mengeksekusi blok kode spesifik dan mengabaikan blok kode yang lain.

1) Statement if

Pernyataan *if* akan menentukan sebuah pernyataan (atau blok kode) yang akan eksekusi jika dan hanya jika persyaratan bernilai benar *(true).* Bentuk dari pernyataan if,

Sintaks Perintah If

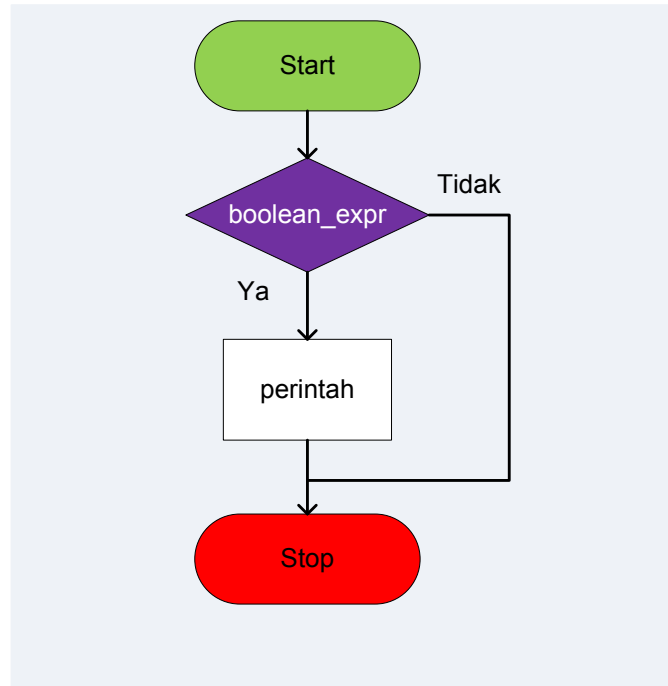
```
if(boolean_expression)

statement;
```

Atau

Sintaks Perintah If

```
if(boolean_expression)
{
      statement1;     statement2;
}
```

dimana, *boolean_expression* adalah sebuah penyataan logika *(true/false)* atau variabel bertipe *boolean*.

Gambar 31. Flowchart Statement if

Petunjuk Penulisan Program:

1. **Boolean_expression** pada pernyataan if harus merupakan nilai boolean). Hal ini berarti persyaratan harus bernilai **true** atau **false**.
2. Masukkan statement didalam blok if. Contohnya,

```
If (boolean_expression){
//statement1;
//statement2;
}
```

2) *Statement if-else*

Pernyataan *if-else* digunakan apabila kita ingin mengeksekusi beberapa pernyataan dengan kondisi *true* dan pernyataan yang lain dengan kondisi *false*. Bentuk statement if-else,

Sintaks Perintah If

```
if(boolean_expression)
statement;
```

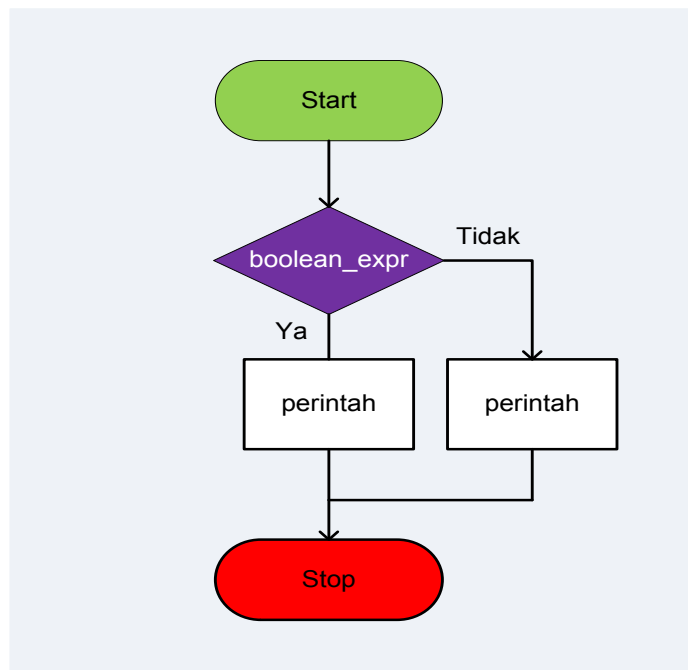
Berikut ini contoh code **statement if-else,**

Listing Program

```
Int grade=68;
if(grade>60)
System.out.println("Congratulations!");
else
System.out.println("Sorry you failed");
```

atau

Listing Program

```
intgrade=68;
if(grade>60)
{
  System.out.println("Congratulations!");
  System.out.println("You passed!");
}
else
{
System.out.println("Sorry you failed");
}
```

Gambar 32.Flowchart Statement If-Else

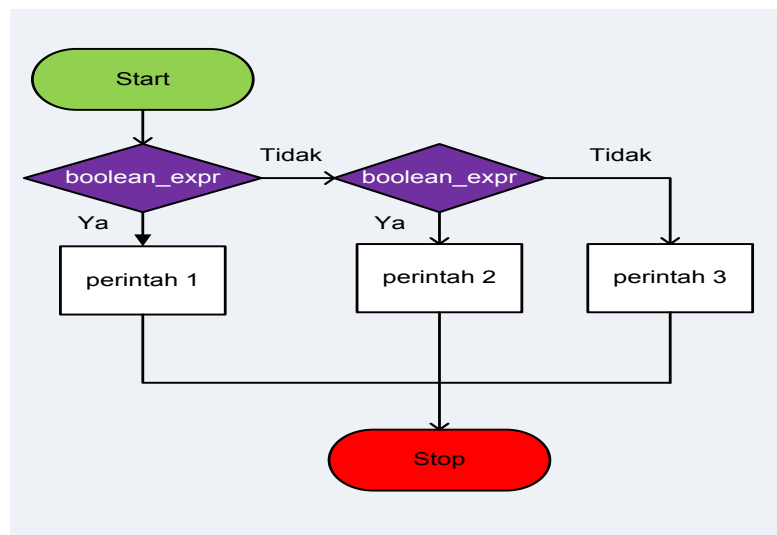
> *Petunjuk Penulisan Program:*
> Untuk menghindari kebingungan, selalu letakkan sebuah pernyataan atau beberapa pernyataan di dalam blokif – else di dalam tanda kurawal {},

3. *Statement if-else-if*

Pernyataan pada bagian kondisi *else* dari blok *if-else* dapat menjadi struktur *if-else* yang lain. Kondisi struktur seperti ini mengijinkan kita untuk membuat seleksi persyaratan yang lebih kompleks. Bentuk statement *if-else if.*

Sintaks perintah If else If

```
if(boolean_expression1)
statement1;
else if(boolean_expression2)
statement2;
else
statement3;
```

Sebagai catatan : anda dapat memiliki banyak blok else-if sesudah pernyataan if. Blok else bersifat opsional dan dapat dihilangkan. Pada contoh yang ditampilkan diatas, jika boolean_expression1 bernilai true, maka program akan mengeksekusi statement1 dan melewati pernyataan yang lain. Jika boolean_expression2 bernilai true, maka program akan mengeksekusi statement2 dan melewati statement2.

Gambar33. Flowchart Statement If-Else-If

Berikut ini contoh code statement if-else-if

Listing Program

```
Int grade=68;
if(grade>90)
{
System.out.println("Very good!");
}
Else if(grade>60)
{
System.out.println("Very good!");
}
else{}
System.out.println("Sorry you failed");
```

4. Statement switch

Cara lain untuk membuat cabang adalah dengan menggunakan kata kunci ***switch***.*Switch* mengkonstruksikan cabang untuk beberapa kondisi dari nilai. Bentuk statement switch adalah sebagai berikut:

Sintaks Perintah Switch

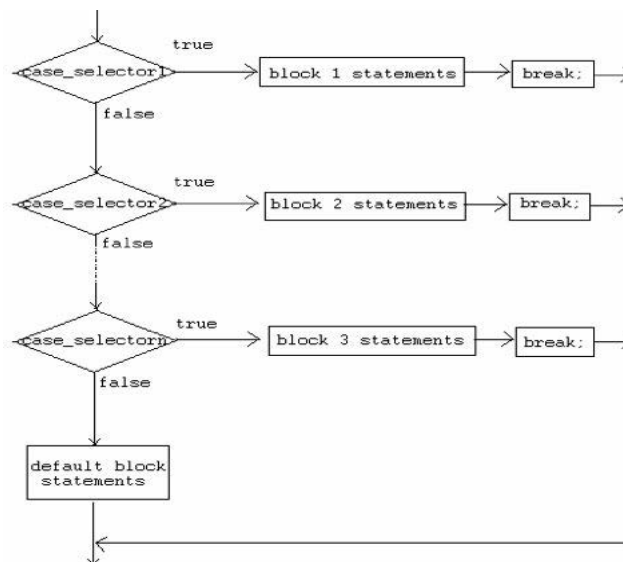
```
switch(switch_expression)
{
Case case_selector1:
statement1;
statement2;
case case_selector2:
statement1;
statement2;
break;
default:
}
statement1;
statement2;
break;
```

Switch_expression adalah ekspresi **integer** atau **karakter** dan *case_selector1*, *case_selector2* dan seterusnya adalah konstanta unik dari nilai *integer* atau karakter. Ketika pernyataan switch ditemukan pada potongan kode program, java pertama kali akan memeriksa *switch_expression*, dan menuju ke *case* yang akan menyamakan nilai yang dimiliki oleh *switch_expression*. Selanjutnya

program akan

mengeksekusi pernyataan pada dari kode setelah *case* yang ditemukan sampai menemui pernyataan *break*, selanjutnya akan mengabaikan pernyataan yang lainnya hingga akhir dari struktur dari pernyataan *switch*.Jika tidak ditemui *case* yang cocok, maka program akan mengeksekusi blok *default*. Sebagai catatan, bahwa bagian blok *default* adalah opsional. Sebuah pernyataan *switch* bias jadi tidak memiliki blok kode *default*.

CATATAN:

- Tidak seperti pada pernyataan ***if***, beberapa pernyataan pada struktur pernyataan *switch* akan dieksekusi tanpa memerlukan tanda kurung kurawal (**{}**).
- Ketika sebuah *case* pada pernyataan *switch* menemui ke cocokan, semua pernyataan pada case tersebut akan dieksekusi. Tidak hanya demikian, pernyataan lain yang berada pada case yang sesuai juga akan dieksekusi.
- Untuk menghindari program mengeksekusi pernyataan pada *case* berikutnya, kita menggunakan pernyataan ***break*** sebagai pernyataan akhir pada setiap blok *case*.

Gambar 34.Flowchart Statement Switch

Petunjuk Penulisan Program:
1. Menentukan penggunaan pernyataan if atau pernyataan switch adalah sebuah keputusan programmer. Programmer dapat menentukan pernyataan yang mana yang akan dipakai berdasarkan kemudahan membaca program dan faktor-faktor yang lain.
2. Pernyataan if dapat digunakan untuk membuat keputusan berdasarkan rentang nilai tertentu atau kondisi tertentu, sedangkan pernyataan switch membuat keputusan hanya berdasarkan nilai unik dari tipe integer atau karakter.

c. Rangkuman

Pernyataan if digunakan untuk membandingkan suatu permasalahan atau objek. If ada bermacam-macam diantaranya, statement if, statement if-else, statement if-else-if, statement switch. Statement if akan mengeksekusi pernyataan hanya jika if bernilai benar atau true. Statement if else digunakan untuk mengeksekusi dua kondisi benar atau salah, true atau false, statement if-else-if bisa digunakan untuk mengeksekusi sebuah kondisi yang lebih dari dua kondisi, dalam statement ini cara penulisan harus berhati-hati. Statement switch hampir sama dengan if-else-if hanya saja swicth mengkonstruksikan cabang untuk beberapa kondisi nilai. Untuk menghentikan program switch menggunakan perintah ***break*** pada case yang sesuai juga akan dieksekusi.

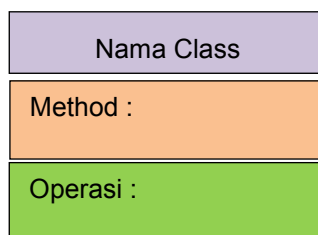
d. Tugas

Tugas 1

Buatlah suatu program yang berfungsi mengecek suatu nilai. Jika nilai lebih dari 75 cetak “Lulus”, jika nilai kurang dari 75 cetak “Gagal”.

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

❖ Bandingkan dan Simpulkan

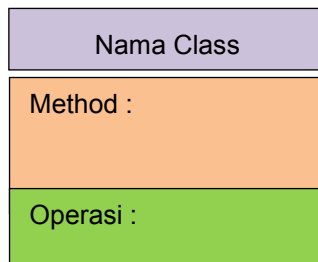
Bandingkan hasil program untuk mengecek nilai yang Anda buat dengan hasil program temanmu. Sebutkan kesimpulan yang ada peroleh!

Tugas 2

Buatlah program pendaftaran siswa baru di sekolah yang memilih jurusan di sekolah tersebut. Terdapat 5 jurusan RPL, Animasi, TKJ, Multimedia, Otomotif. Jika pendaftar memilih salah satu jurusan, cetak nama jurusan yang dipilih.

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ **Bandingkan dan Simpulkan**

Bandingkan hasil program untuk menampilkan nama jurusan yang dipilih yang Anda buat dengan hasil program temanmu. Sebutkan kesimpulan Anda!

e. Test Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa yang dimaksud dengan :
   a. Statement if
   b. Statement if-else
   c. Statement if-else-if
   d. Statement switch
2. Apakah perbedaan menggunakan statement if dan statement switch dalam penulisan program yang sama, dan apakah hasilnya berbeda ?
3. Apakah kelebihan menggunakan statement switch dibandingkan dengan menggunakan statementf if ?
4. Amati program berikut, dan tulislah apa outputnya !

Listing Program

```
public class coba{
public static void main (String [] args){
int a=3;
switch (a) {
     case 1:
        System.out.println("Nilai a=1");
break;
     case 2:
        System.out.println("Nilai a=2"); break;
     case 3:
        System.out.println("Nilai a=3");break;
     default:
        System.out.println("Nilai a=4"); break
          }
     }
}
```

f. Lembar Jawaban Test Formatif (LJ)

LJ- 01 : Apa yang dimaksud dengan :

a) Statement if

....

....

....

....

b) Statement if-else

…..

....

....

....

c) Statement if-else-if

....

....

....

....

d) Statement switch

……..

....

....

....

LJ- 02 : Apakah perbedaan menggunakan statement if dan statement switch dalam penulisan program yang sama, dan apakah hasilnya berbeda ?

....

....

....

....

....

....

....

....

4. Kegiatan Belajar 6 :Dasar dan Aturan Pemrograman Berorientasi Obyek (Perulangan)

a. *Tujuan Pembelajaran*

Setelah mengikuti kegiatan belajar 6 ini siswa diharapkan dapat :

1) Memahami struktur kontrol pengulangan (while, do-while, for)
2) Menggunakan struktur kontrol pengulangan (while, do-while, for) untuk menjalankan blok tertentu pada program beberapa kali

b. *Uraian Materi*

Struktur kontrol pengulangan adalah berupa pernyataan dari Java yang mengijinkan kita untuk mengeksekusi blok code berulang-ulang sesuai dengan jumlah tertentu yang diinginkan. Ada tiga macam jenis dari struktur kontrol pengulangan yaitu *while, do- while, dan for-loops*.

1. whileloop

Pernyataan *whileloop* adalah pernyataan atau blok pernyataan yang diulang-ulang sampai mencapai kondisi yang cocok. Bentuk pernyataan while,

```
while(boolean_expression){ statement1; statement2;}
```

Pernyataan di dalam *whileloop* akan di eksekusi berulang-ulang selama kondisi *boolean_expression* bernilai benar (*true*).

Contoh pada kode dibawah ini.

Listing Program

```
Int i=4;
   while(i>0)
   {
   System.out.print(i);
   i--;
   }
```

Contoh di atas akan mencetak angka 4321 pada layar. Perlu dicatat jika bagian i--; dihilangkan, akan menghasilkan pengulangan yang terus menerus (**infiniteloop**). Sehingga, ketika menggunakan *whileloop* atau bentuk pengulangan yang lain, pastikan Anda memberikan pernyataan yang membuat pengulangan berhenti pada suatu kondisi.

Berikut ini adalah beberapa contoh *whileloop*,

Listing Program

```
intx=0;
   while(x<10)
   {
   System.out.println(x);
   x++;
   }
```

Listing Program

```
//infiniteloop

   while(true) System.out.println("hello");
```

Listing Program

```
//noloops
   //statementisnotevenexecuted while(false)
   System.out.println("hello");
```

2. *do-whileloop*

Do-while loop mirip dengan *while-loop*. Pernyataan di dalam *do-whileloop* akan dieksekusi beberapa kali selama kondisi bernilai benar (*true*). Perbedaan antara *while* dan *do-whileloop* adalah dimana pernyataan di dalam *do-while loop* akan dieksekusi sedikitnya **satu kali.**

Sintaks do-while loop

```
do{
   statement1;
   statement2;
   ...
   }while(boolean_expression);
```

Pernyataan di dalam *do-while loop* akan dieksekusi pertama kali, dan akan dievaluasi kondisi dari *boolean_expression*. Jika nilai pada *boolean_expression* tersebut bernilai *true*, pernyataan di dalam *do-whileloop* akan dieksekusi lagi.

Berikut ini beberapa contoh do-while loop:

Listing Program

```
intx=0;
do
{
System.out.println(x);
x++;
}while(x<10);
```

Contoh ini akan memberikan output 0123456789 pada layar.

Listing Program

```
//infiniteloop
do{
System.out.println(“hello”);
}
while(true);
```

Contoh di atas akan melakukan pengulangan terus menerus yang menulis kata “hello” pada layar.

Listing Program

```
//oneloop
    //statementisexecutedonce do
    System.out.println(“hello”);
    while(false);
```

Contoh di atas akan memberikan output hello pada layar.

Panduan pemrograman:

1. *Kesalahan pemrograman yang biasa terjadi ketika menggunakan do-while loop adalah lupa untuk menulis titik koma(;) setelah ekspresi while.*
   do{...}while(boolean_expression) // salah > tidak ada titik koma (;)
2. *Seperti pada while loop, pastikan do-while loop anda berhenti pada suatu kondisi.*

3. *forloop*

Pernyataan *forloop* memiliki kondisi hampir mirip seperti struktur pengulangan sebelumnya yaitu melakukan pengulangan untuk mengeksekusi kode yang sama sebanyak jumlah yang telah ditentukan.

Bentuk dari forloop,

Sintaks for loop

```
for(InitializationExpression;  LoopCondition;StepExpressio
n)
{
statement1;
statement2;
...
}
```

Dimana ***Initialization Expression***–inisialisasi dari variable loop. ***LoopCondition*** membandingkan variable loop pada nilai batas tertentu. ***Step Expression***- melakukan update pada variable loop.

Berikut ini adalah contoh dari forloop,

Listing Program

```
inti;
for(i=0;i<10;i++)
{
System.out.print(i);
}
```

Pada contoh ini, pernyataan i = 0 merupakan inisialisasi dari variabel. Selanjutnya, kondisi < 10 diperiksa. Jika kondisi bernilai true, pernyataan didalam forloop dieksekusi. Kemudian, ekspresii ++ dieksekusi, lalu akan kembali pada bagian pemeriksaan terhadap kondisii < 10 lagi. Kondisi ini akan dilakukan berulang-ulang sampai mencapai nilai yang salah (false). Contoh tadi, adalah contoh yang sama dari while loop,

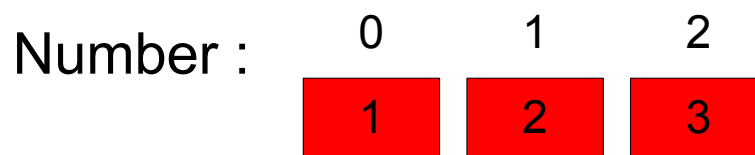
Listing Program

```
inti=0;
while(i<10)
{
System.out.print(i);
i++;
}
```

Pada Bab sebelumnya, kita telah mendiskusikan bagaimana cara pendeklarasian berbagai macam variable dengan menggunakan tipe data primitif. Dalam pendeklarasian variabel, kita sering menggunakan sebuah tipe data beserta nama variabel atau *identifier* yang unik. Apabila kita ingin menggunakan variabel tersebut, kita akan memanggil dengan nama *identifier*-nya.

Sebagai contoh, kita memiliki tiga variabel dengan tipe data int yang memiliki *identifier* berbeda untuk tiap variabel.

```
int number1;
int number2;
int number3;
number1=1;
number2=2;
number3=3;
```

Seperti yang dapat Anda perhatikan pada contoh diatas, kode tersebut akan sia-sia karena harus menginisialisasi dan menggunakan setiap variabel padahal sebenarnya variabel-variabel tersebut digunakan untuk tujuan yang sama. Pada bahasa pemrograman Java maupun dibahasa pemrograman yang lain, terdapat sebuah kemampuan untuk menggunakan satu variable yang dapat menyimpan beberapa data dan memanipulasinya dengan lebih efektif. Tipe variabel inilah yang disebut sebagai **array**.

Gambar 33. Contoh dari Integer Array

Sebuah array akan menyimpan beberapa item data yang memiliki tipe data sama di dalam sebuah blok memori yang berdekatan yang kemudian dibagi menjadi beberapa ruang. Array adalah sebuah variable/sebuah lokasi tertentu yang memiliki satu nama sebagai *identifier*, namun identifier ini dapat menyimpan lebih dari sebuah nilai.

c. Pendeklarasian Array

Array harus dideklarasikan seperti layaknya sebuah variabel. Pada saat mendeklarasikan array, Anda harus membuat sebuah daftar dari tipe data, yang diikuti oleh sepasang tanda kurung [], lalu diikuti oleh nama *identifier*-nya. Sebagai contoh,

Sintaks

```
int[]ages;
```

Atau Anda dapat menempatkan sepasang tanda kurung [] sesudah nama *identifier*. Sebagai contoh,

Sintaks

```
Int ages[];
```

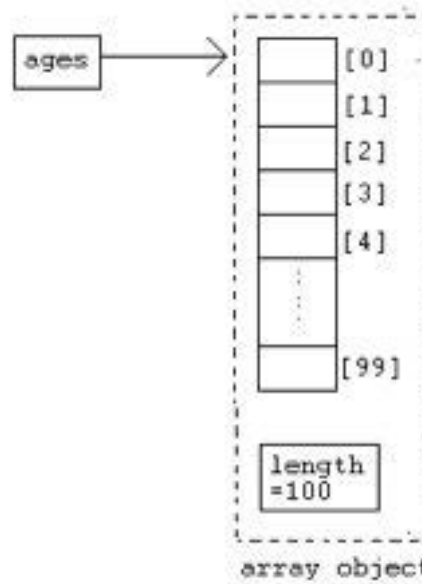
Setelah pendeklarasian array, kita harus membuat array dan menentukan berapa panjangnya dengan sebuah konstruktor. Proses ini di Java disebut sebagai *instantiation* (istilah dalam Java yang berarti membuat). Untuk meng-instantiate sebuah obyek, kita membutuhkan sebuah konstruktor. Kita akan membicarakan lagi mengenai instantiate obyek dan pembuatan konstruktor pada bagian selanjutnya. Sebagai catatan bahwa ukuran dari array tidak dapat diubah setelah anda menginisialisasinya. Sebagai contoh,

Sintaks Deklarasi Array

```
//deklarasi
Int ages[];
//instantiate obyek
ages=new int[100];
```

Atau bisa juga ditulis dengan,

Sintaks Deklarasi Array

```
//deklarasi dan instantiate obyek
Int ages[]=new int[100];
```

Gambar 35. Inisialisasi Array

Pada contoh di atas, pendeklarasian tersebut akan memberitahukan kepada compiler Java, bahwa identifier ages akan digunakan sebagai nama array yang berisi data bertipe integer, dan dilanjutkan dengan membuat atau meng-*instantiate* sebuah array baru yang terdiri dari 100 elemen. Selain menggunakan sebuah pernyataan *new* untuk meng-instantiate array, Anda juga dapat mmendeklarasikan, membangun, kemudian memberikan sebuah nilai pada array sekaligus dalam sebuah pernyataan.

Sebagai contoh,

Listing Program

```
//membuat sebuah array yang berisi variabel-variabel
//Boolean pada sebuah identifier.Array ini terdiri dari
4
//elemen yang diinisilisasikan sebagai value
//{true,false,true,false}
Boolean results[]={true,false,true,false};
//Membuat sebuah array yang terdiri dari
penginisialisasian
//4 variabel double bagi value{100,90,80,75}
```

```
double[]grades={100,90,80,75};

//Membuat sebuah array String dengan identifier
days.Array

//ini terdiri dari 7 elemen.

String
days[]={“Mon”,“Tue”,“Wed”,“Thu”,“Fri”,“Sat”,“Sun”};
```

d. Pengaksesan sebuah elemen array

Untuk mengakses sebuah elemen dalam array, atau mengakses sebagian dari array , Anda harus menggunakan sebuah angka atau yang disebut sebagai **indeks** atau subscript. Pada saat memasukkan nilai ke dalam array, sebuah nomor indeks atau subscript anggota array, sehingga program dan programmer pada array apabila dibutuhkan. Nilai indeks selalu dalam tipe integer, dimulai dari angka nol dan dilanjutkan ke angka berikutnya sampai akhir array. Sebagai catatan bahwa indeks didalam array dimulai dari 0 sampai dengan (ukuranArray-1).Sebagai contoh, pada array yang kita deklarasi kan tadi, kita mempunyai,

Sintaks Elemen Array

```
//memberikan nilai 10 kepada elemen pertama array

ages[0]=10;

//mencetak elemen array yang terakhir

System.out.print(ages[99]);
```

Perlu diperhatikan bahwa sekali array dideklarasikan dan dikonstruksi, nilai yang disimpan dalam setiap anggota array akan diinisialisasi sebagai nol. Oleh karena itu, apabila Anda menggunakan tipe data seperti String, array tidak akan diinisalisasi menjadi string kosong. Untuk itu Anda tetap harus membuat String array secara eksplisit. Berikut ini adalah contoh kode untuk mencetak seluruh elemen di dalam array. Dalam contoh ini digunakanlah pernyataan *forloop*, sehingga kode kita menjadi lebih pendek.

Listing Program

```
Public class Array Sample
{
Public static void main (String[]args)
{
int[]ages=new int[100];
for(int i=0;i<100;i++)
{
System.out.print(ages[i]);
}
}
}
```

Petunjukpenulisanprogram:

1. Biasanya, lebih baik menginisialisasi atau menginstantiate array setelah Anda mendeklarasikannya. Sebagai contoh pendeklarasiannya
   int[]arr=newint[100]; lebih disarankan daripada,
   int[]arr;
   arr=new int[100];
2. Elemen-elemen dalam n-elemen array memiliki indeks dari 0 sampain-1. Perhatikan di sini bahwa tidak ada elemen array arr [n]. Hal ini akan menyebabkan array-index out-of-bounds exception.
3. Anda tidak dapat mengubah ukuran dari sebuah array

e. Array Multidimensi

Array multidimensi diimplementasikan sebagai array yang terletak di dalam array. Array multidimensi dideklarasikan dengan menambahkan jumlah tanda kurung setelah nama array. Sebagai contoh,

Sintaks Array Multidimensi

```
Elemen512x128dariintegerarray int[][]twoD=newint[512][128];
//karakter array 8x16x24
char[][][]threeD=new char[8][16][24];
//String array4 baris x 2 kolom
String[][]dogs=
{
{"terry","brown"},
{"Kristin","white"},
```

```
{"toby","gray"},
{"fido","black"}
};
```

Untuk mengakses sebuah elemen di dalam array multidimensi, sama saja dengan mengakses array satu dimensi. Misalnya saja, untuk mengakses elemen pertama dari baris pertama didalam array dogs, kita akan menulis,

Sintaks Array Multidimensi

```
System.out.print(dogs[0][0]);
```

Kode diatas akan mencetak String “terry” dilayar.

c. Rangkuman

Java mengijinkan kita untuk mengeksekusi blok code berulang-ulang sesuai dengan jumlah yang diinginkan. Ada pernyataan while loop, yaitu pernyataan yang diulang-ulang sampai mencapai kondisi yang diinginkan jika nilai dari while loop bernilai benar atau true maka akan terus berulang-ulang hingga nilai menjadi false atau salah. Do-while-loop mengeksekusi pernyataan di dalam do-while-loop akan dieksekusi sedikitnya satu kali. For loop hampir mirip seperti struktur pengulangan untuk mengeksekusi kode yang sama sebanyak jumlah yang telah ditentukan. Array adalah suatu tipe variabel yang memiliki kemampuan untuk menyimpan beberapa data dan memanipulasinya dengan lebih efektif. Array memiliki suatu nama sebagai identifier, namun identifier ini dapat menyimpan lebih dari sebuah nilai. Tipe variabel ini harus dideklarasi layaknya variabel lain, pada saat mendeklarasikan array anda harus membuat sebuah daftar dari tipe data, yang diikuti oleh sepasang tanda kurung [ ].Sekali array dideklarasikan dan di konstruksi, nilai yang disimpan dalam setiap anggota array akan di inisialisasi sebagai nol. Untuk mengakses array anda harus menggunakan sebuah angka atau yang disebut sebagai indeks atau subscript. Nilai indeks ini selalu bertipe integer dan dimulai dari angka nol dilanjutkan ke angka berikutnya. Sedangkan array multidimensi diimplementasikan seagai array yang terletak di dalam array, untuk mengakses sebuah elemen sama dengan mengakses array satu dimensi.

d. Tugas

Tugas 1

Buatlah program untuk mencetak angka genap mulai dari 1 – 20.

❖ Mengamati Listing Program dan Output Program

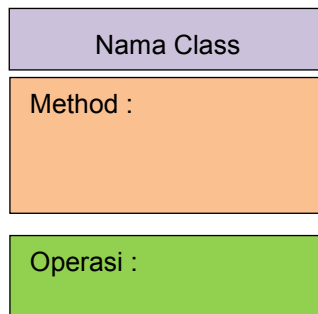
1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

Tugas 2

Tulislah suatu program yang membaca sepuluh bilangan integer dan kemudian menampilan nilai terkecil dan terbesar dari sepuluh integer tersebut.

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan

3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	

9.

10.

e. Test Formatif

Dalam test ini setiap andaharus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Sebutkan 3 statement yang dimiliki oleh struktur kontrol!
2. Jelaskan fungsi dari kata kunci *break* dan *continue* pada perulangan atau looping!
3. Bagaimanakah sintaks mendeklarasikan dan menciptakan sebuah array ?
4. Dapatkah baris-baris di dalam suatu array dua dimensi memiliki panjang yang berbeda ?

f. Lembar Jawaban Test Formatif (LJ)

LJ- 01 : Sebutkan dan jelaskan secara singkat 3 statement yang dimiliki olehstruktur kontrol :

a) ....
....
....
....
b) ....
....
....
....
c)..

LJ- 02 : Jelaskan fungsi dari kata kunci *break* dan *continue* pada perulangan atau looping.

....
....

....

....

....

....

....

....

....

LJ- 03 : Bagaimanakah sintaks mendeklarasikan dan menciptakan sebuah array?

....

....

....

....

....

....

....

....

....

....

LJ- 04: Dapatkah baris-baris di dalam suatu array dua dimensi memiliki panjang yang berbeda?

....

....

....

....

....

....

....

5. Kegiatan Belajar 7 :Konsep Class dan Obyek

a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 7 ini siswa diharapkan dapat :

1) Memahami perbedaan class dan obyek
2) Menyajikan pembuatan class

b. Uraian Materi

1) Perbedaan *Class* dan Obyek

Pada dunia perangkat lunak, sebuah obyek adalah sebuah komponen perangkat lunak yang stukturnya mirip dengan obyek pada dunia nyata. Setiap obyek dibangun dari sekumpulan data (atribut) yang disebut variable untuk menjabarkan karakteristik khusus dari obyek, dan juga terdiri dari sekumpulan method yang menjabarkan tingkah laku dari obyek. Bisa dikatakan bahwa obyek adalah sebuah perangkatlunak yang berisi sekumpulan variable dan method yg berhubungan.Variabel dan method dalam obyek Java secara formal diketahui sebagai variabel instance dan method instance.Hal ini dilakukan untuk membedakan dari variable class dan method class, dimana akan dibahas kemudian.

Class adalah sturktur dasar dari OOP. *Class* terdiri dari dua tipe dari anggota dimana disebut dengan *field* (attribut/properti) dan method. *Field* merupakan tipe data yang didefinisikan oleh *class*, sementara *method* merupakan operasi. Sebuah obyek adalah sebuah *instance* (keturunan) dari *class*.

Untuk dapat membedakan antara *class* dan obyek, mari kita mendiskusikan beberapa contoh berikut ini. Kita memiliki sebuah *class* mobil dimana dapat digunakan untuk medefinisikan beberapa obyek mobil. Pada tabel dibawah, mobil A dan mobil B adalah obyek dari *class* mobil. *Class* memiliki *field* nomor, plat, warna, manufaktur dan kecepatan yang diisi dengan nilai pada obyek mobil A dan mobil B. Mobil juga dapat berakselerasi, berbelok dan melakukan rem.

Tabel 10. *Class* Car dan Obyek-Obyeknya

Class mobil		Obyek mobilA	ObyekMobilB
Variabel Instance	Nomor Plat	ABC111	XYZ123
	Warna	Biru	Merah
	Manufaktur	Mitsubishi	Toyota
	Kecepatan	50km/h	100km/h
Method Instance	*Method* Akselerasi		
	Method Belok		
	Method Rem		

Ketika diinisialisi, setiap obyek mendapat satu set variabel yang baru. Bagaimanapun, implementasi dari *method* dibagi diantara obyek pada *class* yang sama.

Class menyediakan keuntungan dari *reusability*. Programmer perangkat lunak dapat menggunakan sebuah kelas beberapa kali untuk membuat banyak obyek.

✔ *Instansiasi Class*

Untuk membuat sebuah obyek atau sebuah *instance* pada sebuah class. Kita menggunakan operator **new**. Sebagai contoh, jika anda ingin membuat *instance* dari *class string*, kita menggunakan kode berikut:

```
String  str2=new String(“Hello world!”);
```

Ini juga sama dengan,

```
String str2= "Hello";
```

✔ *Variabel Class dan Variabel Method*

Selain dari variabel *instance*, kita juga memungkinkan untuk mendefinisikan variabel dari *class*, yang nantinya variabel ini dimiliki oleh *class*. Ini berarti variabel ini dapat memiliki nilai yang sama untuk semua obyek pada *class* yang sama. Mereka juga disebut *static member variables*.

2) Pembuatan *Class*

Sebelum menulis *class* Anda, pertama pertimbangkan dimana Anda akan menggunakan *class* dan bagaimana *class* tersebut akan digunakan. Pertimbangkan pula nama yang tepat dan tuliskan seluruh informasi atau

property yang ingin Anda isi pada *class*. Jangan sampai terlupa untuk menuliskan secara urut *method* yang akan Anda gunakan dalam *class*.

Dalam pendefinisian *class*, dituliskan:

Sintaks Pembuatan Class

```
<modifier>class<name>
{
<attributeDeclaration>*
<constructorDeclaration>*
<methodDeclaration>*
}
```

Dimana :

`<modifier>` adalah sebuah *access modifier*, yang dapat dikombinasikan denganti pe*modifier* lain.

Pada bagian ini, kita akan membuat sebuah *class* yang berisi *record* dari siswa. Jika kita telah mengidentifikasikan tujuan dari pembuatan *class*, maka dapat dilakukan pemberian nama yang sesuai. Nama yang tepat pada *class* ini adalah StudentRecord.

Untuk mendefinisikan *class*, kita tuliskan:

Sintaks Pembuatan Class

```
Public class StudentRecord
{
//area penulisan kode selanjutnya
}
```

dimana,

Public	-	*Class* ini dapat di akses dari luar *package*
Class	-	*Keyword* yang digunakan untuk pembuatan *Class* dalam Java
StudentRecord	-	*Identifier yang menjelaskan class*

3) Deklarasi Atribut

Dalam pendeklarasian atribut,kita tuliskan:

Sintaks Deklarasi Atribut

```
modifier><type><name>[=<default_value>];
```

Langkah selanjutnya adalah mengurutkan atribut yang akan diisikan pada *class.* Untuk setiap informasi, urutkan juga tipe data yang yang tepat untuk digunakan. Contohnya, Anda tidak mungkin menginginkan untuk menggunakan tipe data *integer* untuk nama siswa, atau tipe data *string* pada nilai siswa. Berikut ini adalah contoh informasi yang akan diisikan pada *class* StudentRecord:

```
name          -String
address       -String
age           -Int
mathgrade     -double
englishgrade  -double
sciencegrade  -double
averagegrade  -double
```

✔ *Instance Variable*

Jika kita telah menuliskan seluruh atribut yang akan diisikan pada *class*, selanjut nya kita akan menuliskannya pada kode.Jika kita menginginkan bahwa atribut–atribut tersebut adalah unik untuk setiap *object* (dalam hal ini untuk setiap siswa), maka kita harus mendeklarasikannya sebagai *instance variable*

Sebagai contoh:

Sintaks Deklarasi Atribut

```
Public class StudentRecord

{

Private String name;

Private String address;

Private int age;

Private double mathGrade; private double englishGrade;
private double scienceGrade; private double average;

}
```

Private disini menjelaskan bahwa variabel tersebut hanya dapat diakses oleh *class* itu sendiri. *Object* lain tidak dapat menggunakan variabel tersebut secara langsung. Kita akan membahas tentang kemampuan akses pada pembahasan selanjutnya.

✓ *Class Variable atau StaticVariables*

Disamping *instance variable*, kita juga dapat mendeklarasikan *class variable* atau variabel yang dimiliki *class* sepenuhnya. Nilai pada variabel ini sama pada semua *object* di *class* yang sama. Anggaplah kita menginginkan jumlah dari siswa yang dimiliki dari seluruh *class*, kita dapat mendeklarasikan satu *staticvariable* yang akan menampung nilai tersebut. Kita beri nama variabel tersebut dengan nama studentCount.

Berikut penulisan *staticvariable:*

Sintaks Class Variable

```
Public class StudentRecord
{
//area deklarasi instance variables
Private static int student Count;
//area penulisan kode selanjutnya
}
```

Kita gunakan *keyword : 'static'* untuk mendeklarasikan bahwa variabel tersebut adalah *static*. Maka keseluruhan kode yang dibuat terlihat sebagai berikut:

Sintaks Class Variable

```
Public class StudentRecord
            {
               private   String  name;
               private   String  address;
               private   int     age;
               private   double  mathGrade;
               private   double  englishGrade;
               private   double  scienceGrade;
               Private   double  average;

               Private   static  intstudentCount;
//area penulisan kode selanjutnya
}
```

c. Rangkuman

Obyek adalah sebuah komponen perangkat lunak yang strukturnya mirip dengan objek pada dunia nyata. Dalam bahasa pemrograman bisa dikatan bahwa objek adalah perangkat lunak yang berisi sekumpulan variabel dan method yang berhubungan. *Class* adalah struktur dasar dari *OOP*, *class* terdiri dari dua tipe anggota dimana disebut dengan *field* dan *method. Field* merupakan tipe data yang didefiniskan, sementara *method* merupakan operasi. Untuk membuat *class*, sebelum menulis nama pertimbangkan dulu nama *class* dan dimana *class* tersebut digunakan. Dalam pendeklarasian atribut untuk menggunakan tipe data *integer* untuk nama siswa, atau tipe data *string* pada nilai siswa. Jika anda menginginkan bahwa atribut-atribut tersebut unik, maka dideklarasikan sebagai *instance variable. Class variable* atau *static variable,* variabel ini sama pada semua object di class yang sama. Anda dapat mendeklarasikan satu *static variable* yang akan menampung nilai tersebut.

d. Tugas

Tugas 1

Buatlah listing program dengan kelas Mobil dan obyeknya Pajero.

❖ Mengamati Listing Program dan Output Program

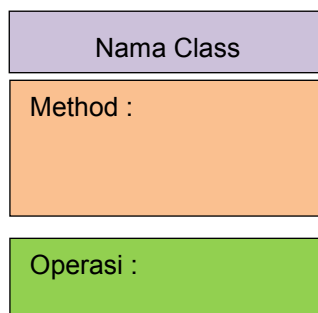
1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

Tugas 2

Buatlah listing program untuk menampilkan behavior macan tutul (obyek) dari kelas binatang buas.

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

e. *Test Formatif.*

Dalam tes ini, Anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa yang dimaksud dengan *Class* ?
2. Apa yang dimaksud dengan Obyek ?
3. Apa perbedaan *instance variabel* dan *static variabel* ?
4. Bagaimanakah sintaks pada pembuatan *class* ?

f. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Apa yang dimaksud dengan Class ?

....
....
....
....
....
....
....

LJ- 02 : Apa yang dimaksud dengan Objek ?

....
....
....
....
....
....
....
....

LJ- 03 : Apa perbedaan instance variabel dan static variabel ?

....
....
....
....
....
....

LJ- 04: Bagaimanakah sintaks pada pembuatan class ?

....
....
....
....
....
....

g. Lembar Kerja Siswa

6. Kegiatan Belajar 8 :Konsep Class dan Obyek

a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 8 ini siswa diharapkan dapat :

1) Memahami *method* dalam *class*
2) Menyajikan penggunaan *method* dalam *class*

b. Uraian Materi

1) *Apakah Method itu dan mengapa menggunakan Method?*

Sebuah *method* adalah bagian-bagian kode yang dapat dipanggil oleh program utama atau dari *method* lainnya untuk menjalankan fungsi yang spesifik.

Berikut adalah karakteristik dari *method*:

- Dapat mengembalikan satu nilai atau tidak sama sekali,
- Dapat diterima beberapa parameter yang dibutuhkan atau tidak ada parameter sama sekali. Parameter bisa juga disebut sebagai argumen dari fungsi,
- Setelah method telah selesai dieksekusi, dia akan kembali pada method yang memanggilnya.

Sekarang mengapa kita butuh untuk membuat banyak *method*? Mengapa kita tidak menuliskan semua kode pada sebuah *method*? Hal ini karena penyelesaian masalah yang sangat efektif adalah memecah masalah-masalah tersebut menjadi beberapa bagian. Kita juga dapat melakukan hal ini di Java dengan membuat *method* untuk mengatasi bagian tertentu dari masalah. Sebuah permasalahan dapat dipecah-pecah menjadi beberapa bagian kecil. Hal ini sangat baik sekali untuk membuat program yang sangat besar.

✓ Memanggil *Instance* dan memberikan Variabel dari *Method*

Sekarang, untuk mengilustrasikan bagaimana memanggil method, mari kita menggunakan *class string* sebagai contoh. Anda dapat menggunakan sebuah dokumentasi dari Java API untuk melihat semua *method* yang tersedia dalam *class string*. Selanjutnya, kita akan membuat *method* kita sendiri. Tapi untuk saat ini, mari terlebih dahulu kita gunakan method yang sudah disediakan oleh Java.

Untuk memanggil sebuah *instance method*, kita dapat menuliskan:

Sintaks Class Variable

```
nameOfObject.nameOfMethod(parameters);
```

Mari kita mengambil dua contoh method yang ditemukan dalam classString.

Tabel 11. Deklarasi *Method*

Deklarasi *method*	Definisi
public charcharAt(intindex).	Mengambil karakter pada indeks
public boolean equalsIgnoreCase (String another String).	Membandingkan antar *String*, tidak *case sensitive*.

✓ *Pemberian Variabel dalam Method*

Pada contoh kita sebelumnya, kita sudah pernah mecoba melewati *variable* pada *method*. Walaupun kita belum dapat membedakan antara perbedaan tipe variabel yang diberikan (*passing*) ke *method* dalam Java. Ada dua tipe data variabel *passing* pada *method*, yang pertama adalah pass-by-value dan yang kedua adalah pass-by-reference.

➢ ***Pass-by-Value***

Ketika pass-by-value terjadi, *method* membuat sebuah salinan dari nilai *variable* yang dikirimkan ke method. Walaupun demikian, *method* tidak dapat secara langsung memodifikasi nilai variabel pengirimnya meskipun parameter salinannya sudah dimodifikasi nilainya di dalam *method*.

➢ ***Pass-by-reference***

Ketika sebuah pass-by-reference terjadi,alamat memori dari nilai pada sebuah variabel dilewatkan pada saat pemanggilan method. Hal ini berarti bahwa *method* menyalin alamat memori dari variabel yang dilewatkan pada *method*. Ini tidak seperti pada pass-by-value, *method* dapat memodifikasi variabel asli dengan menggunakan alamat memori tersebut. Meskipun berbeda nama, variabel yang digunakan dalam *method* dengan variabel aslinya, kedua variabel ini menunjukkan lokasi dari data yang sama.

✓ ***Memanggil Method Static***

Method Static adalah *method* yang dapat dipakai tanpa harus menginisialisasi suatu class (maksudnya tanpa menggunakan variabel terlebih dahulu). *Method static* hanya dimiliki oleh *class* dan tidak dapat digunakan oleh *instance* (atau *object*) dari suatu *class*. *Method static* dibedakan dari *method* yang dapat *instance* di dalam suatu class oleh kata kunci *static*.

Untuk memanggil *method static*, ketikkan kode berikut:

Sintaks pemanggilan method

```
Classname.staticMethodName(params);
```

2) Pembuatan *Method*

Sebelum kita membahas *method* apa yang akan dipakai pada *class*, mari kita perhatikan penulisan *method* secara umum.

Dalam pendeklarasian *method*, kita tuliskan:

Sintaks pembuatan method

```
<modifier><returnType><name>(<parameter>*){
<statement>*
}
```

Dimana,

<modifier>dapat menggunakan beberapa *modifier* yang berbeda

<returnType>dapat berupa seluruh tipe data, termasuk *void*

<name>*identifier* atas *class*

<parameter> ::= <tipe_parameter><nama_parameter>[,]

✓ ***Accessor Method***

Untuk mengimplementasikan enkapsulasi, kita tidak menginginkan sembarang *object* dapat mengakses data kapan saja. Untuk itu, kita deklarasikan atribut dari *class* sebagai *private*. Namun, adakalanya dimana kita menginginkan *object* lain untuk dapat mengakses data *private*. Dalam hal ini kita gunakan *accessor method.*

Accessor Method digunakan untuk membaca nilai variabel pada *class*, baik berupa *instance* maupun *static*. Sebuah *accessor method* *s*umumnya dimulai dengan penulisan ***get <namaInstanceVariable>***. *Method* ini juga mempunyai sebuah *return value*.

Sebagai contoh, kita ingin menggunakan *accessor method* untuk dapat membaca nama, alamat, nilai bahasa Inggris, Matematika, dan ilmu pasti dari siswa.

Mari kita perhatikan salah satu contoh implementasi *accessor method*.

Listing Program

```
Public class StudentRecord
{
Private String name;
Public String getName(){
Return name;
}
}
```

Dimana,

public - Menjelaskan bahwa *method* tersebut dapat diakses dari *object* luar class

String - Tipe data *return value* dari *method* tersebut bertipeString

getName - Nama dari *method*

() - Menjelaskan bahwa *method* tidak memiliki parameter apapun

- ✓ ***Mutator Method***

Bagaimana jika kita menghendaki *object* lain untuk mengubah data? Yang dapat kita lakukan adalah membuat *method* yang dapat memberi atau mengubah nilai variabel dalam *class*, baik itu berupa *instance* maupun *static*. *Method* semacam ini disebut dengan *mutator method*. Sebuah *mutator method* umumnya tertulis **set<namaInstanceVariabel>**. Mari kita perhatikan salah satu dari implementasi *mutator method*:

Listing Program

```
Public class Student Record
{
Private String name;
:
:
Public void setName(String temp){
name=temp;
}
}
```

Dimana,

public - Menjelaskan bahwa *method* ini dapat dipanggil *object* luarclass

void - *Method* ini tidak menghasilkan *return value*

setName - Nama dari *method*

(Stringtemp) - Parameter yang akan digunakan pada *method*

Pernyataan berikut:

name=temp;

mengidentifikasi nilai dari *temp* sama dengan *name* dan mengubah data pada *instance variable name*. Perlu diingat bahwa *mutator methods* tidak menghasilkan *return value*. Namun berisi beberapa argumen dari program yang akan digunakan oleh *method*.

- ✓ ***Multiple Return Statements***

  Anda dapat mempunyai banyak *return values* pada sebuah *method* selama mereka tidak pada blok program yang sama. Anda juga dapat menggunakan konstanta disamping variabel sebagai *return value*. Sebagai contoh, perhatikan *method* berikut ini:

Listing Program

```
Public String getNumberInWords(int num){ String
default Num="zero";
if(num==1){
return"one";//mengembalikan sebuah konstanta
```

```
}
Else if(num==2){
return"two";//mengembalikan sebuah konstanta
}
//mengembalikan sebuah variabel
Return default Num;
}
```

- ✓ *Static Methods*

  Kita menggunakan *static method* untuk mengakses *static variable* studentCount.

Listing Program

```
Public class StudentRecord
{
Private static int studentCount;

Public static int getStudentCount(){
Return studentCount;
}
}
```

dimana,

public	- Menjelaskan bahwa *method* ini dapat diakses dari *object arclass*.
static	- *Method* ini adalah *static* dan pemanggilannya menggunakan **[namaclass].[namaMethod]**.

Sebagai contoh:

`studentRecord.getStudentCount`

Int	- Tipe *return* dari *method.* Mengindikasikan *method* tersebut harus mempunyai *return value* berupa integer.
public	- Menjelaskan bahwa *method* ini dapat diakses dari *object* luar *class.*
getStudentCount()	- Nama dari *method.*
	- *Method* ini tidak memiliki parameter apapun.

Pada deklarasi diatas, *method* getStudentCount() akan selalu menghasilkan *return value* 0 jika kita tidak mengubah apapun pada kode program untuk mengatur nilainya. Kita akan membahas pengubahan nilai dari studentCount pada pembahasan *constructor*.

Petunjuk Penulisan Program:

1. *Nama method harus dimulai dengan huruf kecil*
2. *Nama method harus berupa kata kerja*
3. *Gunakan dokumentasi sebelum mendeklarasikan sebuah method. Anda dapat Menggunakan Java Doc.*

c. Rangkuman

Method merupakan bagian-bagian kode yang dapat dipanggil oleh program utama atau dari *method* lainya. Anda dapat menggunakan dokumentasi dari Java API untuk melihat semua *method* yang tersedia dalam class string. Pemberian variabel dalam *method* terdapat dua tipe data variabel *passing* pada method, yang pertama adalah pass-by-value dan yang kedua adalah pass-by-reference. Pass-by-value membuat sebuah salinan dari nilai variabel yang dikirimkan ke *method*, namun tidak dapat secara langsung memodifikasi nilai variabel pengirimnya. Pass-by-reference *method* menyalin alamat memori dari variabel yang dilewatkan pada *method*, *method* dapat memodifikasi variabel asli dengan menggunakan alamat memori tersebut. Dalam pembuatan *method* terdapat *accesor method* yang berfungsi mengimplementasikan enkapsulasi sehingga kita tidak menginginkan sembarang *object* dapat mengakses data kapan saja. Sedangkan *method mutator* adalah *method* yang dapat memberi atau mengubah nilai variabel dalam class, baik itu berupa *instance* maupun *static.* Untuk mengakses *static variable* kita menggunakan *static method.*

d. Tugas

Tugas 1

Tulislah suatu metode untuk menghitung penjumlahan digit dalam suatu integer.

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan

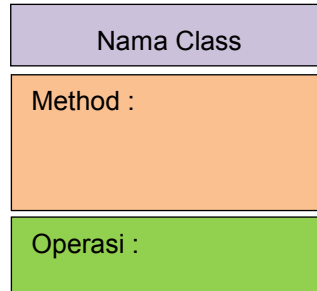
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

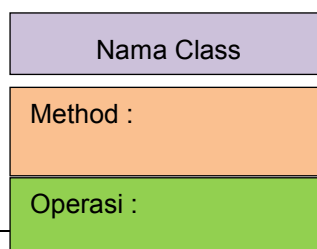
❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

Tugas 2

Tulislah suatu metode untuk menampilkan matrik **n x n**. Setiap elemen 0 atau 1, dibangkitkan secara acak. Tulislah suatu program untuk menguji dan menampilkan suatu matriks 3 x 3.

❖ Mengamati Listing Program dan Output Program

1. Mengamati bentuk matriks.
2. Menentukan nama Class
3. Menentukan variabel yang digunakan
4. Menentukan nama Method
5. Gambar Class Diagram

Nama Class
Method :
Operasi :

6. Buatlah listing program
7. Compile dan debug program

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ **Bandingkan dan Simpulkan**

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

e. *Test Formatif.*

Dalam tes ini Anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas, tulislah jawabannya pada lembar jawaban *test* formatif yang telah disediakan.

1. Apakah yang membedakan antara *accessor method* dan *mutator method* ?
2. Apa perbedaan konstruktor dan metode ?
3. Sebutkan dan jelaskan macam-macam pemberian variabel dalam *method*.
4. Apa yang dimaksud dengan *method* ? dan berikan contohnya.

f. *Lembar Jawaban Test Formatif (LJ).*

LJ- 01 : Apakah yang membedakan antara *accessor method* dan *mutator method*?

....
....
....
....
....
....
....
....
....

LJ- 02 : Apa perbedaan konstruktor dan metode ?

....
....
....
....
....
....
....
....

7. Kegiatan Belajar 9 :Konsep Class dan Obyek

a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 9 ini siswa diharapkan dapat :

1) Memahami penggunaan referensi *this*
2) Menyajikan penggunaan *constructor* dalam *class*

b. Uraian Materi

1) *Reference this*

Reference this digunakan untuk mengakses *instance variable* yang dibiaskan oleh parameter. Untuk pemahaman lebih lanjut, mari kita perhatikan contoh pada *method set Age*. Dimisalkan kita mempunyai kode deklarasi berikut pada *method set Age*.

Sintaks *reference this*

```
Public void setAge(intage) {
age=age;//SALAH!!!
}
```

Nama parameter pada deklarasi ini adalah *age*, yang memiliki penamaan yang sama dengan *instance variable age*. Parameter *age* adalah deklarasi terdekat dari *method*, sehingga nilai dari parameter tersebut akan digunakan. Maka pada pernyataan

```
age=age;
```

kita telah menentukan nilai dari parameter age kepada parameter itu sendiri. Hal ini sangat tidak kita kehendaki pada kode program kita. Untuk menghindari kesalahan semacam ini, kita gunakan metode referensi ***this***. Untuk menggunakan tipe referensi ini, kita tuliskan:

```
this.<namaInstanceVariable>
```

Sebagai contoh,kita dapat menulis ulang kode hingga tampak sebagai berikut:

Sintaks *reference this*

```
publicvoidsetAge(intage){
this.age=age;
}
```

Method ini akan mereferensikan nilai dari parameter *age* kepada *instance variable* dari *object* StudentRecord.

CATATAN : Anda hanya dapat menggunakan referensi this terhadap *instance variable* dan BUKAN *static* ataupun class variabel

Variabel kelas (*class variable*) adalah variabel yang dideklarasikan di dalam sebuah kelas dan bertidak sebagai data *field* dari kelas tersebut, sedangkan variabel lokal (*local variable*) adalah variabel yang dideklarasikan di dalam sebuah metoda. Cakupan variabel lokal dimulai dari posisi variabel tersebut dideklarasikan sampai dengan akhir dari blok metoda yang ditandai dengan *closing brace*. Cakupan dari variabel kelas meliputi keseluruhan kelas. Pemberian nama yang sama antara variabel kelas dan variabel lokal di sebuah metoda mungkin saja bisa terjadi. Misalnya, metoda *set* yang digunakan untuk merubah nilai variabel kelas, mungkin saja mendeklarasikan parameter dengan nama sama sebagaimana nama variabel kelas yang nilainya akan dirubah.

Di dalam metoda *set*, untuk dapat mengacu ke variabel kelas yang nilainya akan dirubah, Anda perlu menggunakan kata kunci *this*. Apabila variabel kelas tersebut dideklarasikan menggunakan *modifier static*, maka variabel kelas dapat diakses menggunakan *nama-kelas.variabel-static*. Di dalam contoh program kelas Warna, terdapat dua variabel kelas yaitu variabel merah dan biru. Kelas Warna juga mendeklarasikan dua buah metoda yaitu metoda setMerah yang memiliki parameter merah dan setBiru yang memiliki parameter biru. Parameter dari kedua metoda tersebut memiliki kesamaan nama dengan dua variabel kelas (*data field*) dari kelas Warna. Berikut ini adalah kode program kelas Warna yang mungkin dapat memberikan pemahaman lebih jelas tentang penggunaan kata kunci (*keyword*) *this* untuk mengacu ke variabel kelas.

Listing Program

```
// Deklarasi kelas
class Warna {
// Deklarasi variabel kelas (data field)
int merah = 7;
static double biru = 2;

// Deklarasi metoda
```

```
void setMerah(int merah) {
this.merah = merah;
         }
// Deklarasi metoda static
static void setBiru( double biru) {
               Warna.biru = biru;
         }
}
```

Misalnya c1 adalah variabel acuan yang mengacu ke objek dari kelas Warna. Memanggil metoda dengan pernyataan c1.setMerah(5) sama dengan mengeksekusi c1.merah = 5, dimana kata kunci *this* diganti dengan c1. Baris nomor 10 adalah pernyataan pemberian yaitu memberikan nilai parameter merah ke data *field* merah dari objek pemanggil (misalnya c1). Sedangkan di baris nomor 15, pernyataan tersebut mempunyai arti bahwa nilai di parameter biru diberikan ke data *field* statik biru dari kelas Warna.

2) Construtor

Telah tersirat pada pembahasan sebelumnya, *Constructor* sangatlah penting pada pembentukan sebuah object. *Constructor* adalah *method* dimana seluruh inisialisasi object ditempatkan.

Berikut ini adalah *property* dari Constructor:

- *Constructor* memiliki nama yang sama dengan *class*
- Sebuah *Constructor* mirip dengan *method* pada umumnya, namun hanya informasi–informasi berikut yang dapat ditempatkan pada *header* sebuah *constructor*, *scope* atau identifikasi pengaksesan (misal:public), nama dari konstuktor dan parameter.
- *Constructor* tidak memiliki *return value*
- *Constructor* tidak dapat dipanggil secara langsung, namun harus dipanggil dengan menggunakan operator ***new*** pada pembentukan sebuah *class*.

Untuk mendeklarasikan *constructor*, kita tulis,

Sintaks mendeklarasikan *consturctor*

```
<modifier><className>(<parameter>*){
<statement>*
}
```

✓ ***Default Constructor***

Setiap *class* memiliki default constructor. Sebuah *default constructor* adalah *constructor* yang tidak memiliki parameter apapun. Jika sebuah *class* tidak memiliki *constructor* apapun, maka sebuah *default constructor* akan dibentuk secara implicit.

Sebagai contoh, pada *class* StudentRecord, bentuk *default constructor* akan terlihat seperti dibawah ini :

```
Public StudentRecord()
{
        //area penulisan kode
}
```

✓ Menggunakan *Constructor*

Untuk menggunakan *constructor*, kita gunakan kode–kode sebagai berikut:

Listing Program

```
Public static void main(String[]args)

              {

//membuat3objek

            StudentRecord
            annaRecord=newStudentRecord("Anna");

            StudentRecord
            beahRecord=newStudentRecord("Beah","Philippine
            s"); StudentRecord
            crisRecord=newStudentRecord(80,90,100);

            //area penulisan kode selanjutnya

              }
```

Sebelum kita lanjutkan, mari kita perhatikan kembali deklarasi variabel *static* studentCount yang telah dibuat sebelumnya.Tujuan deklarasi studentCount adalah untuk menghitung jumlah *object* yang dibentuk pada *class* StudentRecord. Jadi, apa yang akan kita lakukan selanjutnya adalah menambahkan nilai dari studentCount setiap kali setiap pembentukan *object* pada *class* StudentRecord. Lokasi yang tepat untuk memodifikasi

dan menambahkan nilai studentCounter letak pada constructor-nya, karena selalu dipanggil setiap kali obyek terbentuk. Sebagai contoh:

Listing Program

```
Public StudentRecord()
{
                    //letak kode inisialisasi
                    studentCount++;//menambahstudent
               }

Public StudentRecord(Stringtemp)
{
                    this.name=temp;
                    studentCount++;//menambahstudent
               }

Public StudentRecord(Stringname,Stringaddress)
{
                    this.name=name;
                    this.address=address;
                    studentCount++;//menambahstudent
               }

publicStudentRecord(doublemGrade,doubleeGrade,doubles
Grade)
{
                    mathGrade=mGrade;
                    englishGrade=eGrade;
                    scienceGrade=sGrade;
                    studentCount++;//menambahstudent
               }
```

✓ Pemanggilan *Constructor* Dengan *this()*

Pemanggilan *constructor* dapat dilakukan secara berangkai, dalam arti Anda dapat memanggil *constructor* di dalam *constructor* lain. Pemanggilan dapat dilakukan dengan referensi *this()*.

Perhatikan contoh kode sebagai berikut:

Listing Program

```
public StudentRecord(){
this("some string");
}
public StudentRecord(String temp){
this.name = temp;
}
public static void main( String[] args )
{
StudentRecord annaRecord = new StudentRecord();
}
```

Dari contoh kode diatas, pada saat baris ke 13 dipanggil akan memanggil *constructor* dasar pada baris pertama. Pada saat baris kedua dijalankan, baris tersebut akan menjalankan *constructor* yang memiliki parameter String pada bariske-6.

Beberapa hal yang patut diperhatikan pada penggunaan ***this()*** :

- Harus dituliskan pada baris pertama pada sebuah *constructor,*
- Hanya dapat digunakan pada satu definisi *constructor*. Kemudian metode ini dapat diikuti dengan kode–kode berikutnya yang relevan.

c. Rangkuman

Referensi *this* digunakan untuk mengakses *instance* variable yang dibiaskan oleh parameter. Variabel kelas (*class variable*) adalah variabel yang dideklarasikan di dalam sebuah kelas dan bertindak sebagai data *field* dari kelas tersebut, sedangan variabel lokal (*local variable*) adalah variabel yang dideklarasikan di dalam sebuah metoda. *Constructor* sangatlah penting pada pembentukan sebuah *object*. *Constructor* adalah *method* dimana seluruh inisialisasi object ditempatkan. *Default constructor* adalah *constructor* yang tidak memiliki parameter apapun. Pemanggilan *constructor* dapat

dilakukan secara berangkai, dalam arti Anda dapat memanggil *constructor* di dalam *constructor* lain dengan menggunakan referensi *this()*.

d. Tugas

Tugas 1

Buatlah listing program untuk menampilkan nama dan nim mahasiswa. Gunakan nama kelas Siswa, dan gunakan referensi **this** untuk mengakses instance variabel.

❖ Mengamati Listing Program dan Output Program

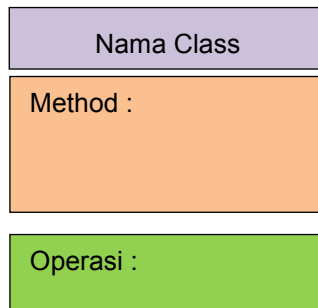
1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

Tugas 2

Buatlah listing program untuk menampilkan nilai sebuah volume Balok dengan menggunakan prinsip konstruktor (pada saat deklarasi obyek Balok, setiap obyeknya disertai nilai panjang, lebar, tinggi).

❖ Mengamati Listing Program dan Output Program

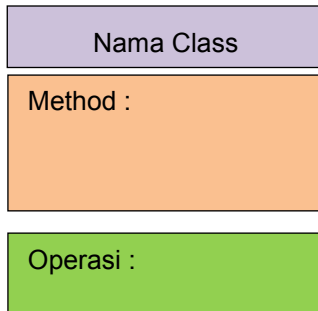
1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	

9.

10.

❖ **Bandingkan dan Simpulkan**

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

e. *Test Formatif*

Dalam test ini Anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apakah fungsi dari kata kunci *this* ?
2. Sebutkan property dari *Constructor* !
3. Apa saja yang perlu diperhatikan dalam penggunaan kata kunci *this* ?

f. *Lembar Jawaban Test Formatif (LJ)*

LJ- 01 : Apakah fungsi dari kata kunci this ?

....

....

....

....

LJ- 02 : Sebutkan property dari Constructor !

....

....

....

....

....

....

....

....

8. Kegiatan Belajar 10 :Pembungkusan Data

a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 10 dan 11 ini siswa diharapkan dapat:

1) Memahami konsep enkapsulasi
2) Menerapkan konsep enkapsulasi dalam *class*

b. Uraian Materi

1) Enkapsulasi dan *modifier*

Enkapsulasi merupakan teknik yang membuat variabel/*field class* menjadi bersifat *private* dan menyediakan akses ke variabel/*field* melalui *public method*. Jika *field* di deklarasikan sebagai *private*, maka *field* ini tidak bisa diakses oleh siapapun diluar *class*, dengan demikian *field* disembunyikan di dalam *class*.

Manfaat utama teknik enkapsulasi adalah kita mampu memodifikasi kode tanpa merusak kode yang telah digunakan pada *class* lain. Enkapsulasi memiliki manfaat sebagai berikut:

- ✓ Modularitas

  Source code dari sebuah *class* dapat dikelola secara independen dari *source code class* yang lain. Perubahan internal pada sebuah *class* tidak akan berpengaruh bagi *class* yang menggunakannya.

- ✓ *Information Hiding*

  Penyembunyian informasi yang tidak perlu diketahui objek lain.

Pada saat membuat, mengatur *properties* dan *class method*, kita ingin untuk mengimplementasikan beberapa macam larangan untuk mengakses data. Sebagai contoh, jika Anda ingin beberapa atribut hanya dapat diubah hanya dengan *method* tertentu, tentu Anda ingin menyembunyikannya dari obyek lain pada *class*. Di Java, implementasi tersebut disebut dengan *access modifiers*.

2) Penerapan enkapsulasi dalam *class*

Kita dapat menyembunyikan information dari suatu *class* sehingga anggota-anggota *class* tersebut tidak dapat diakses dari luar. Adapun caranya adalah cukup dengan memberikan akses *control private* ketika mendeklarasikan suatu atribut atau *method*. Contoh:

```
private int nrp;
```

Encapsulation (Enkapsulasi) adalah suatu cara untuk menyembunyikan implementasi detail dari suatu class. Enkapsulasi mempunyai dua hal mendasar, yaitu:

- ✓ *information hiding*
- ✓ menyediakan suatu perantara (*method*) untuk pengaksesan data

Contoh:

Listing Program

```
public class Siswa {
private int nrp;
public void setNrp(int n) {
nrp=n;
}
}
```

Contructor (konstruktor) adalah suatu *method* yang pertama kali dijalankan pada saat pembuatan suatu obyek. Konstruktor mempunyai ciri yaitu:

- ✓ mempunyai nama yang sama dengan nama class,
- ✓ tidak mempunyai return type (seperti void, int, double, dan lain-lain).

Contoh:

Listing Program

```
public class Siswa {
private int nrp;
private String nama;
public Siswa(int n, String m) {
nrp=n;
nama=m;
}
}
```

Suatu *class* dapat mempunyai lebih dari 1 konstruktor dengan syarat daftar parameternya tidak boleh ada yang sama.

Contoh:

Listing Program

```
public class Siswa {
private int nrp;
private String nama;
public Siswa(String m) {
nrp=0;
nama="";
}
public Siswa(int n, String m) {
nrp=n;
nama=m;
}
}
```

c. Rangkuman

Enkapsulasi merupakan teknik yang membuat variabel/*field class* menjadi bersifat *private* dan menyediakan akses ke variabel/*field* melalui. Manfaat utama teknik *encapsulation* adalah kita mampu memodifikasi kode tanpa merusak kode yang telah digunakan pada *class* lain. Di Java, implementasi tersebut disebut dengan *access modifiers*.

d. Tugas

Tugas 1

Buatlah program untuk menghitung gaji bersih dari seorang pegawai, pajak ppn sebesar 10% dari gaji kotor.

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

e. Test Formatif

Dalam test ini setiap andaharus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa yang dimaksud enkapsulasi bidang data ?
2. Apakah keuntungan enkapsulasi bidang data ?

f. Lembar Jawaban Test Formatif (LJ)

LJ- 01 : Apa yang dimaksud enkapsulasi bidang data ?

....

....

....

....

....

LJ- 02 : Apakah keuntungan enkapsulasi bidang data ?

....

....

....

9. Kegiatan Belajar 11: Pembungkusan

a. *Tujuan Pembelajaran*

Setelah mengikuti kegiatan belajar 13 ini siswa diharapkan dapat :

1) Memahami konsep pewarisan
2) Menciptakan *superclass* dan *subclass*

b. *Uraian Materi*

Terdapat 4 macam *access modifiers* di JAVA, yaitu : *public*, *private*, *protected* dan *default*. 3 tipe akses pertama tertulis secara ekplisit pada kode untuk mengindikasikan tipe akses, sedangkan yang keempat yang merupakan tipe default, tidak diperlukan penulisan *keyword* atas tipe.

No	Modifier	Class	Paket Sama		Paket Berbeda	
			Extend	Instan	Extend	Instan
1	Public	√	√	√	√	√
2	Protected	√	√	√	√	
3	Default	√	√	√		
4	Private	√				

✓ ***Public***

Dapat dilihat pada table diatas bahwa keyword Public dapat diakses didalam class itu sendiri, dapat diakses dengan menggunakan metode *extend* dan instan pada paket yang sama, serta dapat diakses dengan metode *extend* maupun instan dalam paket yang berbeda. Artinya hak akses *public* dapat diakses oleh sembarang object manapun dan dimanapun posisinya serta dengan apapun caranya.

Data maupun *method* yang bersifat *public* dapat diakses oleh semua bagian didalam program. Dengan kata lain, data–data maupun *method-method* yang dideklarasikan dengan tingkat akses *public* akan dikenali atau dapat diakses oleh semua kelas yang ada didalam, baik yang merupakan kelas turunan maupun kelas yang tidak memiliki hubungan sama sekali. Untuk mendeklarasikan suatu data atau method dengan tingkat akses *public*, gunakan kata kunci *public*.

Berikut contoh program sederhana :

Listing Program

```
class atas
{
public int a;
protected int b;
private int c;
}
class bawah{
public static void main(String[]args){
atas objek = new atas();
objek.a=2;
objek.b=3;
System.out.println("nilai a: "+objek.a);
System.out.println("nilai b: "+objek.b);
}
}
```

program diatas akan menghasilkan tampilan berikut:

```
nilai a: 2
nilai b: 3
```

program diatas terdiri dari dua kelas yaitu kelas sekunder yang berisi variabel a, b dan c dengan tingkat akses yang berbeda, dan kelas primer yang berisi objek untuk melakukan *instance* pada kelas turunan, objek pada kelas primer hanya dapat mengisi nilai pada variabel a dan b karena kedua variabel tersebut memiliki tingkat akses *public* dan *protected*, karena variabel c memiliki tingkat akses *private* maka obyek pada kelas primer tidak bisa mengisi variabel tersebut.

✓ ***Protected***

Suatu data maupun *method* yang dideklarasikan dengan tingkat akses *protected* dapat diakses oleh kelas yang memilkinya dan juga oleh kelas–kelas yang masih memiliki oleh hubungan turunan. Sebagai contoh, apabila data x dalam kelas A dideklarasikan sebagai *protected*, maka kelas B (yang merupakan turunan dari kelas A) diizinkan untuk mengakses data x. Namun apabila terdapat kelas lain, misalnya C (yang bukan merupakan turunan dari kelas A maupun B), tetap tidak dapat mengakses data – data yang

dideklarasikan dengan tingkat akses *protected*. Untuk mendeklarasikan suatu data atau *method* dengan tingkat akses *protected*, gunakan kata kunci *ptotected*.

Listing Program

```
public class motor
{
    protected String jenismotor;
    protected String address;
    public motor()
        program turunan:
        program honda.java
        public class honda extends motor
        {
            protected String jenishonda;
            protected String kecepatanhonda;
            public honda()
            {
```

dari contoh program protected yang dapat mengakses hanya kelas motor dan kelas turunannya, yaitu Honda

- ✓ ***Private***

  Dengan mendeklarasikan data dan method menggunakan tingkat akses private, maka data dan *method* tersebut hanya dapat diakses oleh kelas yang memilikinya saja. Ini berarti data dan *method* tersebut tidak boleh diakses atau digunakan oleh kelas-kelas lain yang terdapat didalam program. Untuk mendeklarasikan suatu data atau *method* dengan tingkat akses *private*, gunakan kata kunci *private*.

Listing Program

```
public class Siswa
{
//akses dasar terhadap variabel
private String nama;
//akses dasar terhadap metode
private String getNama(){
return name;
}
}
```

Pada contoh diatas, variabel nama dan *method* getNama() hanya dapat diakses oleh *method internal class* tersebut.

✓ ***Default***

Untuk hak akses *default* ini, sebenarnya hanya ditujukan untuk *class* yang ada dalam satu paket, atau istilahnya hak akses yang berlaku untuk satu folder saja (tidak berlaku untuk *class* yang tidak satu folder/package).

Listing Program

```
public class Siswa{
//akses dasar terhadap variabel
String nama;

//akses dasar terhadap method
String getName(){
return nama;
}
}
```

Pada contoh diatas, variabel nama dan *method* getNama() hanya dapat diakses oleh *method internal class* tersebut.

c. Rangkuman

Terdapat 4 macam *access modifiers* di JAVA, yaitu : *public*, *private*, *protected* dan *default*. *Public* dapat diakses di dalam *class* itu sendiri, dapat diakses dengan menggunakan metode *extend* dan instan pada paket yang sama, serta dapat diakses dengan metode *extend* maupun instan dalam paket yang berbeda. *Protected* dapat diakses oleh kelas yang memilikinya dan juga oleh kelas-kelas yang masih memiliki oleh hubungan turunan. Sedangkan *private*, maka data dan *method* tersebut hanya dapat diakses oleh kelas yang memilikinya saja. *Default* sebenarnya hanya ditujukan untuk *class* yang ada dalam satu paket, atau istilahnya hak akses yang berlaku untuk satu folder saja (tidak berlaku untuk *class* yang tidak satu folder/*package*).

d. Tugas

Tugas 1

Buatlah program untuk menghitung gaji bersih dari seorang pegawai, pajak ppn sebesar 10% dari gaji kotor.

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

e. Test Formatif

Dalam tes ini Anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban *test formatif* yang telah disediakan.

1. Apa yang anda pahami terkait *keyword private*, *protected*, *public*?
2. Apa yang terjadi jika anda membuat sebuah *property* atau *method* menjadi *private*, *protected*, *public* ?

f. Lembar Jawaban Test Formatif (LJ)

LJ- 01 : Apa yang anda pahami terkait keyword *private*, *protected*, *public*?

....

....

....

....

....

....

....

LJ- 02 : Apa yang terjadi jika anda membuat sebuah property atau method menjadi private, protected, public ?

....

....

....

....

....

....

....

....

....

....

10. Kegiatan Belajar 12 : Pewarisan

a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 13 ini siswa diharapkan dapat :

1) Memahami konsep pewarisan
2) Menciptakan superclass dan subclass

b. Uraian Materi

1) Konsep Inheritas

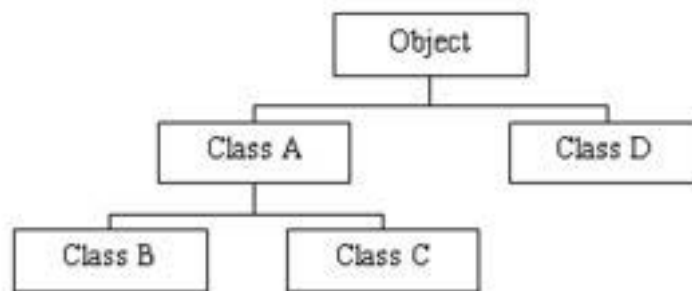
Konsep *inheritance* ini mengadopsi dunia riil dimana suatu entitas/obyek dapat mempunyai entitas/obyek turunan. Dengan konsep inheritance, sebuah class dapat mempunyai class turunan.Suatu class yang mempunyai class turunan dinamakan parent class atau base class. Sedangkan class turunan itu sendiri seringkali disebut subclass atau child class. Suatu subclass dapat mewarisi apa-apa yang dipunyai oleh parent class. Karena suatu subclass dapat mewarisi apa apa yang dipunyai oleh parent class-nya, maka member dari suatu subclass adalah terdiri dari apa-apa yang ia punyai dan juga apa-apa yang ia warisi dari class parent-nya. Kesimpulannya, boleh dikatakan bahwa suatu subclass adalah tidak lain hanya memperluas (extend) parent class-nya.

Gambar 36. Mamalia

Dari hirarki diatas dapat dilihat bahwa, semakin kebawah, class akan semakin bersifat spesifik. Class mamalia memiliki seluruh sifat yang dimiliki oleh binatang, demikian halnya juga macan , kucing, Paus dan Monyet memiliki seluruh sifat yang diturunkan dari class mamalia. Dengan konsep ini, karakteristik yang dimiliki oleh class binatang cukup didefinisikan didefinisikan dalam class binatang saja.

Class mamalia tidak perlu mendefinisikan ulang apa yang telah dimiliki oleh class binatang, karena sebagai class turunannya, ia akan mendapatkan karakteristik dari class binatang secara otomatis. Demikian juga dengan class macan, kucing, Paus dan monyet, hanya perlu mendefinisikan karakteristik yang spesifik dimiliki oleh class-nya masing-masing.Dengan

memanfaatkan konsep pewarisan ini dalam pemrograman, maka hanya perlu mendefinisikan karakteristik yang lebih umum akan didapatkan dari class darimana ia diturunkan.

Dalam Java, semua class, termasuk class yang membangun Java API, adalah subclasses dari superclass Object. Contoh hirarki class diperlihatkan di bawah ini. Beberapa class di atas class utama dalam hirarki class dikenal sebagai superclass. Sementara beberapa class di bawah class pokok dalam hirarki class dikenal sebagai sub class dari class tersebut.

Gambar 37. Hierarki Class di Java

Pewarisan adalah keuntungan besar dalam pemrograman berbasis object karena suatu sifat atau method didefinisikan dalam *superclass*, sifat ini secara otomatis diwariskan dari semua *subclasses*. Jadi, Anda dapat menuliskan kode method hanya sekali dan mereka dapat digunakan oleh semua subclass. Subclass hanya perlu mengimplementasikan perbedaannya sendiri dan induknya.

Konsep inheritance ini mengadopsi dunia riil dimana suatu entitas/obyek dapat mempunyai entitas/obyek turunan. Dengan konsep inheritance, sebuah class dapat mempunyai class turunan. Suatu class yang mempunyai class turunan dinamakan parent class atau base class. Sedangkan class turunan itu sendiri sering kali disebut subclass atau child class. Suatu subclass dapat Mewari siapa-apa yang dipunyai oleh parent class-nya, sehingga member dari suatu subclass adalah terdiri dari apa-apa yang ia punyai dan juga apa-apa yang diawarisi dari classparent-nya. Kesimpulannya, boleh dikatakan bahwa suatu subclass adalah tidak lain hanya memperluas (extend) parentclass-nya.

✓ Kapan menerapkan inheritas?

Kita baru perlu menerapkan inheritance pada saat kita jumpai ada suatu class yang dapat diperluas dari class lain.

Misal terdapat class Pegawai public class Pegawai {public String nama;public double gaji;} Misal terdapat class Manager public class Manajer {public String nama;public double gaji;public String departemen;} Dari 2 buah class diatas, kita lihat class Manajer mempunyai data member yang identik sama dengan class Pegawai, hanya saja ada tambahan data member departemen.Sebenarnya yang terjadi disana adalah class Manajer merupakan perluasan dari class Pegawai dengan tambahan data member departemen.

Disini perlu memakai konsep inheritance, sehingga class Manajer dapat kita tuliskan seperti berikut :

public class Manajer extends Pegawai {public String departemen;}

✓ Keuntungan inheritas

- Subclass menyediakan state/behaviour yang spesifik yang membedakannya dengan superclass, hal ini akan memungkinkan programmer Java untuk menggunakan ulang source code dari superclass yang telah ada.
- Programmer Java dapat mendefinisikan superclass khusus yang bersifat generik, yang disebut abstract class, untuk mendefinisikan class dengan behaviour dan state secara umum.

✓ Deklarasi inheritas

Di dalam Java untuk mendeklarasikan suatu class sebagai sub class dilakukan dengan cara menambahkan kata kunci extends setelah deklarasi nama class, kemudian diikuti dengan nama parentclass-nya. Kata kunci extends tersebut memberitahu compiler Java bahwa kita ingin melakukan perluasan class.

Berikut adalah contoh deklarasi inheritance:

```
public class B extends A{
.....
}
```

Contoh di atas memberitahukan compiler Java bahwa kita ingin meng-extend class A ke class B. Dengan kata lain, class B adalah subclass

(class turunan) dari class A, sedangkan class A adalah parent class dari class B.

Java hanya memperkenankan adanya single inheritance. Konsep single inheritance hanya memperbolehkan suatu sublass mempunyai satu parent class. Dengan konsep single inheritance ini, masalah pewarisan akan dapat diamati dengan mudah.

Dalam konsep dasar inheritance dikatakan bahwa suatu subclass adalah tidak lain hanya memperluas (extend) parentclass-nya. Contoh:

Listing Program

```
Public class Pegawai{
Public String nama;
Public double gaji;
        }
Public class Manajer extends Pegawai{
Public String departemen;
        }
```

Pada saat class Manajer menurunkan atau memperluas (extend) class Pegawai, maka ia mewarisi data member yang dipunyai oleh class Pegawai. Dengan demikian, class Manajer mempunyai data member yang diwarisi oleh Pegawai (nama, gaji), ditambah dengan data member yang ia punyai (departemen).

- ✓ Kontrol Pengaksesan

Pengaksesan member yang ada di parent class dari subclass-nya tidak jauh berbeda dengan pengaksesan member subclass itu sendiri. Contoh:

Suatu parent class dapat tidak mewariskan sebagian member-nya kepada subclass-nya. Sejauh mana suatu member dapat diwariskan ke class lain, ataupun suatu member dapat diakses dari class lain, sangat berhubungan dengan access control (control pengaksesan). Di dalam java, kontrol pengaksesan dapat digambarkan dalam table berikut ini:

Tabel 12. Kontrol Pengaksesan Class

Modifier	Class yang sama	package yang sama	subclass	Class manapun
Private	√			
Default	√	√		
Protected	√	√	√	
Public	√	√	√	√

c. *Rangkuman*

Konsep inheritance ini mengadopsi dunia riil dimana suatu entitas/obyek dapat mempunyai entitas/obyek turunan. Dengan konsep inheritance, sebuah class dapat mempunyai class turunan. Di dalam Java untuk mendeklarasikan suatu class sebagai subclass dilakukan dengan cara menambahkan kata kunci extends setelah deklarasi nama class, kemudian diikuti dengan nama parent class-nya.

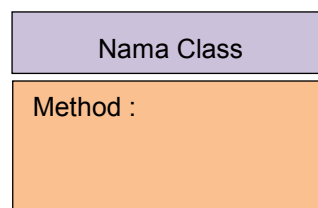
d. *Tugas*

Tugas 1

Buatlah sebuah super class dengan nama transport yang di dalamnya akan kita buat method bernama Kendaraan yang mencetak Kendaraan punya roda, stang, rem, dan jok. Selanjutnya buatlah sub class Mobil beserta method atau karakteristiknya, lalu tampilkan method Mobil beserta method-nya sebagai sebuah alat transport.

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :

Operasi :

5. Buatlah listing program
6. Compile dan debug program

❖ **Bandingkan dan Simpulkan**

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

Tugas 2

Buatlah kelas induk Hewan dan kelas anak Herbivora beserta masing-masing method-nya. Kemudian tampilkan karakteristik/method Hewan Herbivora tersebut.

❖ **Mengamati Listing Program dan Output Program**

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class

Method :

Operasi :

5. Buatlah listing program
6. Compile dan debug program

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

e. Test Formatif

Dalam test ini setiap andaharus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa arti keyword extends dan implements, kapan menggunakannya ?
2. Apa yang dimaksud dengan inheritance ?

f. Lembar Jawaban Test Formatif (LJ)

LJ- 01 : Apa arti keyword extends dan implements, kapan menggunakannya

....
....
....
....
....
....
....
....
....
....

LJ- 02 : Apa yang dimaksud dengan inheritance ?

....
....
....
....
....

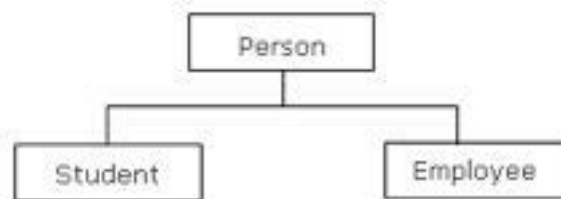
11. Kegiatan Belajar 13: Pewarisan

a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 16 siswa diharapkan dapat :

1) Memahami arti Superclass dan Subclass
2) Mendefinisikan Superclass dan Subclass

b. Uraian Materi

Mendefinisikan Superclass dan Subclass untuk memperoleh suatu class,kita menggunakan kata kunci **extend**.Untuk mengilustrasikan ini, kita akan membuat contoh class induk. Dimisalkan kita mempunyai class induk yang dinamakan Person.

Listing Program

```
public class Person
{
protected String name;
protected String address;
public Person(){
System.out.println("Inside Person:Constructor");
name = "";
address = "";
}
public Person(String name,String address ){
this.name = name;
this.address = address;
}
public String getName(){
return name;
}
public String getAddress(){
```

```
return address;
}
public void setName( String name ){
this.name = name;
}
public void setAddress( String add ){
this.address = add;
}
}
```

Perhatikan bahwa atribut *name* dan *address* dideklarasikan sebagai **protected**. Alasannya kita melakukan ini yaitu, kita inginkan atribut-atribut ini untuk bisa diakses oleh sub classes dari super classess. Jika kita mendeklarasikannya sebagai private, sub classes tidak dapat menggunakannya. Catatan bahwa semua properti dari superclass yang dideklarasikan sebagai **public, protected dan default** dapat diakses oleh sub classes-nya.

Sekarang, kita ingin membuat class lain bernama Student. Karena Student juga sebagai Person, kita putuskan hanya meng-*extend* class Person, sehingga kita dapat mewariskan semua property dan method dari setiap class Person yang ada. Untuk melakukan ini kita tulis,

Listing Program

```
public class Student extends Person
{
public Student(){
System.out.println("Inside Student:Constructor");
//beberapa kode di sini
}
}
```

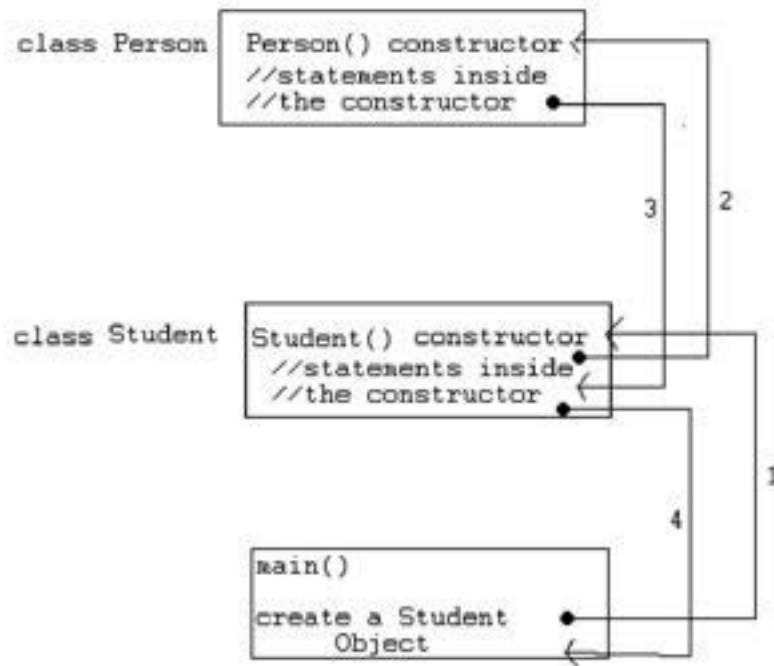
Ketika object Student di-*instantiate*, default constructor dari super class secara mutlak meminta untuk melakukan inisialisasi yang seharusnya. Setelah itu, pernyataan di dalam sub class dieksekusi.

Untuk mengilustrasikannya, perhatikan kode berikut,

Listing Program

```
public static void main(String[]args )
{
Student anna = new Student();
}
```

Dalam kode ini,kita membuat sebuah object dari class Student. Keluaran dari program adalah,

```
Inside Person:Constructor
Inside Student:Constructor
```

Gambar 38. Alur Program

c. Rangkuman

Dalam konsep dasar inheritance dikatakan bahwa suatu sub class adalah tidak lain hanya memperluas (extend) parent class-nya. Pengaksesan member yang ada diparent class dari sub class-nya tidak jauh berbeda dengan pengaksesan member sub class itu sendiri. semua properti dari super class yang dideklarasikan sebagai public, protected dan default dapat diakses oleh sub classes-nya.

d. Tugas

Tugas 1

Buatlah program untuk menampilkan luas permukaan dan volume tabung. Gunakan parent-class Luas Lingkaran (method: jari-jari).

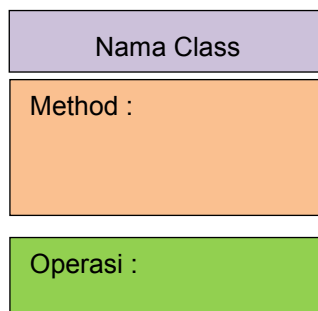
❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

5. Buatlah listing program
6. Compile dan debug program

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

Tugas 2

Buatlah program untuk menampilkan karakteristik (bentuk paruh, makanan, warna bulu, dan bentuk tungkai) Elang dari Kelas Burung.

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	

7.

8.

9.

10.

❖ **Bandingkan dan Simpulkan**

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

e. Test Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa perbedaan superclass dan subclass ?
2. Apa yang anda ketahui tentang interface di java ?

f. Lembar Jawaban Test Formatif (LJ)

LJ- 01 : Apa perbedaan superclass dan subclass

a. Superclass :

....

....

....

....

....

g. Lembar Kerja Siswa

12. Kegiatan Belajar 14 : Pewarisan

a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 14 siswa diharapkan dapat :

1) Memahami penggunaan kata kunci super
2) Menerapkan penggunaan kata kunci cuper dalam inheritas
3) Memahami konsep overloading dan overriding
4) Menyajikan overloading dan overriding dalam class

b. Uraian Materi

1) Kata kunci super

Subclass juga dapat memanggil constructor secara eksplisit dari superclass terdekat. Hal ini dilakukan dengan pemanggil constructor **super**. Pemanggil constructor super dalam constructor dari subclass akan menghasilkan eksekusi dari superclass constructor yang bersangkutan, berdasar dari argument sebelumnya. Sebagai contoh, pada contoh class sebelumnya. Person dan Student, kita tunjukkan contoh dari pemanggil constructor super. Diberikan kode berikut untuk Student,

Sintaks kunci super

```
public Student(){
super( "SomeName", "SomeAddress" );
System.out.println("Inside Student:Constructor");
}
```

Kode ini memanggil constructor kedua dari superclass terdekat (yaitu Person) dan mengeksekusinya. Contoh kode lain ditunjukkan sebagai berikut,

Listing Program

```
public Student(){
super();
System.out.println("Inside Student:Constructor");
}
```

Kode ini memanggil default constructor dari superclass terdekat (yaitu Person) dan mengeksekusinya. Ada beberapa hal yang harus diingat ketika menggunakan pemanggil constuktor super:

a. Pemanggil super() **harus dijadikan pernyataan pertama dalam** constructor.
b. Pemanggil super() hanya dapat digunakan dalam definisi constructor.
c. Termasuk constructor this() dan pemanggil super() **tidak boleh terjadi dalam** constructor **yang sama**.

Pemakaian lain dari supe radalah untuk menunjuk anggota dari superclass (seperti reference **this**). Sebagai contoh,

Listing Program

```
public Student()
{
super.name = "somename";
super.address = "some address";
}
```

c. *Rangkuman*

Subclass juga dapat memanggil constructor secara eksplisit dari superclass terdekat. Hal ini dilakukan dengan pemanggil constructor super. Pemanggil constructor super dalam constructor dari subclass akan menghasilkan eksekusi dari superclass constructor yang bersangkutan. Pemanggil super() hanya dapat digunakan dalam definisi constructor. Termasuk constructor this() dan pemanggil super() tidak boleh terjadi dalam constructor yang sama. Pemakaian lain dari super adalah untuk menunjuk anggota dari superclass (seperti reference **this**).

d. Tugas

Tugas 1

Buatlah program untuk menghitung volume kubus dengan mengambil method dari class Persegi dengan memanfaatkan prinsip inheritas. Gunakan constructor **this()**.

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

5. Buatlah listing program
6. Compile dan debug program

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ Bandingkan dan Simpulkan
Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

e. Test Formatif

Dalam test ini setiap andaharus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apakah fungsi dari kata kunci super ?
2. Apa yang harus diingat ketika menggunakan pemanggil constuktor super ?

f. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Apakah fungsi dari kata kunci super ?

....
....
....
....
....
....
....
....

LJ- 02 : Apa yang harus diingat ketika menggunakan pemanggil constuktor super ?

....
....
....
....
....
....
....
....
....

13. Kegiatan Belajar 15 :Pewarisan

a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 14 siswa diharapkan dapat :

1) Memahami konsep overloading dan overriding
2) Memahami metode final
3) Menyajikan overloading dan overriding dalam class

b. Uraian Materi

1) Metode Overloading

Overloading adalah suatu keadaan dimana beberapa method sekaligus dapat mempunyai nama yang sama, akan tetapi mempunyai fungsionalitas yang berbeda. Overloading ini dapat terjadi pada class yang sama atau pada suatu parent class dan subclass-nya. Overloading mempunyai ciri-ciri sebagai berikut:

- ✓ Nama method harus sama
- ✓ Daftar parameter harus berbeda
- ✓ Return type boleh sama, juga boleh berbeda

Contoh penggunaan overloading dilihat di bawah ini:

`Gambar(intt 1)`	→	1parametertitik,untuk menggambartitik
`Gambar(intt 1, int t2)`	→	2parametertitik,untukmenggambargaris
`Gambar(intt 1, int t2, int t3)`	→	3parametertitik,untukmenggambarsegitiga
`Gambar(intt 1, int t2, int t3, int t4)`	→	3parametertitik,untukmenggambarsegiempat

2) Overriding Method

Overriding adalah suatu keadaan dimana method pada subclass menolak method pada parent class-nya. Overriding mempunyai ciri-ciri sebagai berikut:

- ✓ Nama method harus sama
- ✓ Daftar parameter harus sama

- ✓ Return type harus sama

Untuk beberapa pertimbangan, terkadang class asal perlu mempunyai implementasi berbeda dari method yang khusus dari *superclass* tersebut. Oleh karena itulah, method overriding digunakan. *Subclass* dapat mengesampingkan method yang didefinisikan dalam *superclass* dengan menyediakan implementasi baru dari method tersebut. Misalnya kita mempunyai implementasi berikut untuk method getName dalam superclass Person,

Listing Program

```
public class Person
{
:
public String getName()
{
System.out.println("Parent: getName");
return name;
}
  }
```

Untuk override, method getName dalam subclass Student, kita tulis,

Listing Program

```
public class Student extends Person
{
public String getName()
{
System.out.println("Student: getName");
return name;
}
}
```

Jadi,ketika kita meminta method getName dari object class Student, method overridde akan dipanggil, keluarannya akan menjadi,

```
Student: getName
```

3) Method final dan classfinal

Dalam Java, juga memungkinkan untuk mendeklarasikan class-class yang tidak lama menjadi subclass. Class ini dinamakan **class final**. Untuk mendeklarasikan class untuk menjadi final kita hanya menambahkan kata kunci **final** dalam deklarasi class. Sebagai contoh, jika kita ingin class Person untuk dideklarasikan final, kita tulis,

```
public final class Person
{
      //area kode
}
```

Beberapa class dalam Java API dideklarasikan secara final untuk memastikan sifatnya tidak dapat di-*override*. Contoh-contoh dari class ini adalah Integer , Double , dan String. Ini memungkinkan dalam Java membuat method yang tidak dapat di-*override.* Method ini dapat kita panggil **method final**. Untuk mendeklarasikan method untuk menjadi final,kita tambahkan kata kunci final kedalam deklarasi method. Contohnya, jika kita ingin method getName dalam class Person untuk dideklarasikan final, kita tulis,

Sintaks getName

```
public final String getName(){
return name;
}
```

Method static juga secara otomatis final.Ini artinya Anda tidak dapat membuatnya override.

c. Rangkuman

Overloading adalah suatu keadaan dimana beberapa method sekaligus dapat mempunyai nama yang sama, akan tetapi mempunyai fungsionalitas yang berbeda. Overriding adalah suatu keadaan dimana method pada subclass menolak method pada parent class-nya. *Subclass* dapat mengesampingkan method yang didefinisikan dalam *superclass* dengan menyediakan implementasi baru dari method tersebut. Dalam Java, juga memungkinkan untuk mendeklarasikan class-class yang tidak lama menjadi subclass. Class ini dinamakan **class final**. Untuk mendeklarasikan class untuk menjadi final kita hanya menambahkan kata kunci **final** dalam deklarasi

class. Beberapa class dalam Java API dideklarasikan secara final untuk memastikan sifatnya tidak dapat di-*override*.

d. Tugas

Tugas 1

Buatlah program untuk menampilkan luas segitiga dengan superclass bangun_datar dan sub class Segitiga. Gunakan prinsip overriding dan atau overloading.

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

No	Output Program
1.	
2.	
3.	
4.	

5.

6.

7.

8.

9.

10.

❖ Bandingkan dan Simpulkan
Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

e. Test Formatif.

Dalam test ini setiap andaharus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa arti dari override ?
2. Apa yang dimaksud dengan method overload ?
3. Apa perbedaan Method overriding dan method overload ?
4. Bagaimana anda secara eksplisit memanggil suatu konstruktor superclass dari subclass ?

f. Lembar Jawaban Test Formatif (LJ)

LJ- 01 : Apa arti dari override ?

....

....

....

....

....

g. *Lembar Kerja Siswa.*

14. Kegiatan Belajar 16 :Polimorphisme

a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 16 siswa diharapkan dapat :

1) Memahami konsep polimorfisme
2) Menyajikan overloading dan overriding dalam class

b. Uraian Materi

1) Metode Overloading

Polymorphism merupakan salah satu konsep penting dalam object oriented programming (OOP) khusunya di bahasa Java setelah abstraction dan inheritance. Polymorphism berarti banyak bentuk. Ada beberapa definisi berbeda tentang polymorphism yang berkaitan dengan pemrograman berorientasi obyek. Sedangkan apa yang dimaksud dengan polymorphism sendiri, sebenarnya sulit untuk didefinisikan. Sejalan dengan contoh yang diberikan, Anda diharapkan dapat mengerti dan memahami konsep polymorphism itu sendiri.

Polymorphism sering dikaitkan dengan penggunaan lebih dari satu metoda dengan nama sama. Penggunaan metoda dengan nama sama dapat diterapkan dengan method overloading dan method overriding. Peran polymorphism sebenarnya tidak terbatas hanya pada hal tersebut. Ada keterkaitan antara polymorphism dan inheritance (turunan).

Dalam konsep turunan, saat obyek dari subclass dikonstruksi, obyek dari superclass juga ikut dikonstruksi. Jadi setiap instance dari subclass adalah juga instance dari superclass. Apabila Anda mendeklarasikan metoda dengan parameter dari tipe superclass, Anda diperbolehkan untuk memberi argumen berupa obyek subclass yang merupakan turunan dari superclass tersebut.

Berikut ini adalah contoh program yang dapat memberikan gambaran berkaitan dengan konsep polymorphism. Perlu dipahami dan dimengerti bahwa kelas Object merupakan akar dari semua kelas Java dan menduduki puncak tertinggi dalam hirarkhi. Program akan mendefinisikan kelas yang berkaitan dengan bidang datar secara sederhana termasuk beberapa kelas turunannya (kelas PersegiPanjang dan Balok) dan membatasi hanya pada penerapan method overriding.

Listing Program

```
// Nama file : Polimorphism.java
// Contoh penerapan konsep polimorphism
public class Polimorphism {
   public static void main(String[ ] args) {
      cetakObjek(new Balok());
      cetakObjek(new PersegiPanjang());
      cetakObjek(new BangunDatar());
      cetakObjek(new Object());
   }
   public static void cetakObjek(Object objek) {
      System.out.println(objek);
   }
} // Akhir kelas Polimorphism

class Balok extends PersegiPanjang {
   public String toString() {
      return "Mempunyai sisi panjang, lebar dan
tinggi";
   }
}

class PersegiPanjang extends BangunDatar {
   public String toString() {
      return "Mempunyai sisi panjang dan lebar";
   }
}

class BangunDatar extends Object {
   public String toString() {
      return "Mempunyai berbagai bentuk";
   }
}
```

Baris nomor 14 -16 adalah deklarasi metoda cetakObjek yang mempunyai satu parameter dengan tipe kelas Object. Kelas Object merupakan akar dari semua kelas di Java. Langsung maupun tidak langsung, semua kelas di Java merupakan turunan dari kelas Object. Anda dapat memanggil atau menggunakan metoda cetakObjek dengan argumen berupa obyek yang dibuat dari kelas turunan superclass Object.

Ketika metoda cetakObjek dipanggil (baris nomor 8 – 11), argumen obyek akan diminta. obyek sebagai argumen metoda dapat berupa obyek yang merupakan kelas turunan dari kelas Object yaitu kelas BangunDatar, kelas PersegiPanjang maupun kelas Balok. Masing-masing kelas turunan mendeklarasikan ulang metoda toString yang mempunyai implementasi berbeda. Java Virtual Machine (JVM) akan menentukan secara dinamis implementasi metode toString yang akan digunakan saat program dijalankan. Kemampuan menentukan secara dinamis ini disebut dengan dynamic binding.

Dari gambaran program di atas, apabila parameter sebuah metoda adalah tipe superclass, maka argumen metoda yang diberikan dapat berupa tipe dari subclassnya. Kemampuan seperti inilah yang dimaksud dengan polymorphism. Dari gambaran tersebut, dapat didefinisikan kembali bahwa polymorphism adalah kemampuan untuk menghasilkan sesuatu yang berbeda dengan cara yang sama. Pemberian obyek dari subclass ke obyek dari superclass dapat dilakukan tanpa perlu melakukan konversi.

c. Rangkuman

Polymorphism merupakan salah satu konsep penting dalam object oriented programming (OOP) khusunya di bahasa Java setelah abstraction dan inheritance. Polymorphism berarti banyak bentuk.Polymorphism sering dikaitkan dengan penggunaan lebih dari satu metoda dengan nama sama. Penggunaan metoda dengan nama sama dapat diterapkan dengan method overloading dan method overriding. Pada saat obyek yang sudah dibuat tersebut memanggil overridden method pada parent class, compiler Java akan melakukan invocation (pemanggilan) terhadap overriding method pada subclass dimana yang seharusnya dipanggil adalah overridden method.

d. Tugas

Tugas 1

Buatlah program untuk menampilkan beberapa jenis kendaraan (Mobil, Kereta Api, Pesawat) yang memiliki jenis bahan bakar yang berbeda pula. Gunakan prinsip polimorfisme serta tentukan super class dan sub class-nya.

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

e. *Test Formatif*

Dalam test ini setiap andaharus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa itu polymorphisme ?
2. Jelaskan perbedaan antara *overloading method* dan *overridding method* !

f. Lembar Jawaban Test Formatif (LJ)

LJ- 01 : Apa itu polymorphisme:

....
....
....
....
....
....

LJ- 02 : Apa yang dimaksud dengan polymorphic argument ?

....
....
....
....
....

LJ- 03 : Apa yang dimaksud dengan VirtualMethodInvocation(VMI) ?

....
....
....
....
....
....
....
....

LJ- 04: Jelaskan perbedaan antara *overloading method* dan *overridding method*

....
....
....
....
....
....
....
....

g. Lembar Kerja Siswa.

15. Kegiatan Belajar 17 : Polimorphisme (*Virtual Methode Invocation)*

a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 14 siswa diharapkan dapat :

1) Memahami konsep overloading dan overriding
2) Memahami metode final
3) Menyajikan overloading dan overriding dalam class

b. Uraian Materi

Virtual Method Invocation (VMI) bisa terjadi jika terjadi polimorfisme dan overriding. Pada saat obyek yang sudah di buat tersebut memanggil overridden method pada parent class, compiler Java akan melakukan invocation (pemanggilan) terhadap overriding method pada subclass , dimana yang seharusnya di panggil adalah overridden method. Berikut contoh terjadinya VMI:

Listing Program

```
classParent{
intx=5;
publicvoidInfo(){
System.out.println("IniclassParent");
}
}
classChildextendsParent{
intx=10;
publicvoidInfo(){
System.out.println("IniclassChild");
}
}
publicclassTes{
publicstaticvoidmain(Stringargs[]){
Parenttes=newChild();
System.out.println("Nilaix="+tes.x);
tes.Info();
}
}
```

Hasil dari running program diatas adalah sebagai berikut:

```
Nilaix=5

IniclassChild
```

Polymorphic arguments adalah tipe suatu parameter yang menerima suatu Nilai yang bertipe subclass-nya. Berikut contoh dari polymorphics arguments:

Listing Program

```
Class Pegawai{
}
Class Manajer extends Pegawai{
                     …
}
Public class Tes{
Public static void Proses(Pegawaipeg){
                                             …
}
Public static void main(Stringargs[]){
Manajerman=newManajer(); Proses(man);
}
   }
```

Pernyataan instance of sangat berguna untuk mengetahui tipe asal dari suatu Polymorphic arguments. Untuk lebih jelasnya, misalnya dari contoh program sebelumnya, kita sedikit membuat modifikasi pada class Tes dan ditambah sebuah class baru Kurir, seperti yang tampak dibawah ini:

Listing Program

```
Class Kurir extends Pegawai
}
Public classTes{
publicstaticvoidProses(Pegawaipeg){
if(peginstanceofManajer){
…lakukantugas-tugasmanajer…
}elseif(peginstanceofKurir){
```

```
…lakukantugas-tugaskurir…

}else{
…lakukantugas-tugaslainnya…
}


}
publicstaticvoidmain(Stringargs[]){
Manajerman=newManajer();
Kurirkur=newKurir();
Proses(man);
Proses(kur);
}
}
```

Seringkali pemakaian instance of diikuti dengan casting object dari tipe parameter ke tipe asal. Misalkan saja program kita sebelumnya. Pada saat kita sudah melakukan instance of dari tipe manajer, kita dapat melakukan casting object ke tipe asalnya, yaitu manajer. Caranya adalah seperti berikut:

Listing Program

```
if(peginstanceofManajer)
{
 Manajerman=(Manajer)peg;
…lakukantugas-tugasmanajer…

}
```

c. Rangkuman

Virtual Method Invocation (VMI) terjadi karena objek yang sudah di buat tersebut memanggil overriden method pada parent class dan akan melakukan panggilan pada overriding method yang seharusnya adalah overriden. Polymorphic arguments adalah tipe suatu parameter yang menerima suatu nilai yang bertipe subclass-nya. Pernyataan instance of

sangat berguna untuk mengetahui tipe asal dari suatu polymorphic arguments } Seringkali pemakaian instance of diikuti dengan casting object dari tipe parameter ke tipe asal.

d. Tugas

Tugas 1

Buat program untuk menampilkan nilai siswa, dimana banyaknya siswa tidak ditentukan, kemudian :

a. Tampilkan data dari yang awal hingga akhir
b. Hitung Jumlah nilai seluruh siswa
c. Hitung rata-rata
d. Nilai Maksimum
e. Nilai Minimum

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

5. Buatlah listing program
6. Compile dan debug program

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

e. Test Formatif

Dalam test ini setiap andaharus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa yang dimaksud dengan polymorphic argument ?
2. Apa yang dimaksud dengan VirtualMethodInvocation(VMI) ?

f. Lembar Jawaban Test Formatif (LJ)

LJ- 01 : Apa yang dimaksud dengan polymorphic argument ?

....

....

....

....

....

....

....

....

....

....

LJ- 02 : Apa yang dimaksud dengan VirtualMethodInvocation(VMI) ?

....

....

....

....

....

....

....

....

....

16. Kegiatan Belajar 18 : Polimorphisme (*Casting Objek dan InstanceOf*)

a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 16 siswa diharapkan dapat :

1) Memahami konsep polimorfisme
2) Menyajikan overloading dan overriding dalam class

b. Uraian Materi

Anda telah menggunakan operator casting untuk mengubah variabel-variabel suatu tipe primitif menjadi tipe primitif yang lain. *Casting* dapat pula digunakan untuk mengubah objek dengan suatu tipe kelas menjadi objek degan tipe kelas lain, di dalam suatu hirarki pewarisan. Pada bagian sebelumnya , statemen

```
m(new) Mahasiswa ( ) ) ;
```

menugaskan objek **new Mahasiswa()** kepada suatu parameter bertipe **Object. Statement tersebut ekuivalen dengan**

```
object o = new Mahasiswa(); // Casting implisit

m(o);
```

Statement **Object o = new Mahasiswa()**, dikenal sebagai casting implisit, merupakan hal yang sah karena suatu instans **Mahasiswa** secara otomatis adalah suatu instans **Object**.

Seandainya anda ingin menugaskan referensi objek **o** kepada suatu variabel bertipe **Mahasiswa** menggunakan statemen berikut ini:

```
Mahasiswa b = o;
```

Pada kasus ini, error kompilasi terjadi. Mengapa statemen **Object o = new Mahasiswa()** dapat dilakukan sementara **Mahasiswa b = o** tidak bisa dilakukan ? Alasanya adalah bahwa suatu objek **Mahasiswa** selalu merupakan suatu instans **Object,** tetapi suatu instans **Object** belum tentu merupakan suatu instans **Mahasiswa.** Meskipun anda dapat melihat bahwa **o** adalah suatu objek **Mahasiswa**, kompiler tidak cukup pintar untuk mengetahuinya. Untuk

memberitahu kompiler bahwa **o** merupakan suatu objek **Mahasiswa** , gunakan casting ekspilit. Sintaks yang digunakan sama seperti yang digunakan untuk meng-casting tipe primitif, tipe objek target diapit oleh sepasang kurung dan ditempatkan sebelum objek yang akan dicast:

Mahasiswa b = (Mahasiswa) o; // Casting eksplisit

Adalah hal yang selalu memungkinkan untuk melakukan *casting* terhadap instans dari suatu subkelas menjadi suatu variabel superkelas (yang dikenal dengan *upcasting)*, karena instans subkelas selalu merupakan instans superkelas. Ketika melakukan casting terhadap instans dari suatu superkelas menjadi suatu variabel subkelasnya (yang dikenal dengan *downcasting),* casting eksplisit harus digunakan untuk mengaskan tujuan anda kepada kompiler.

Agar casting berhasil dilakukan , Anda perlu memastikan bahwa objek yang akan di-cast merupakan suatu instans subkelas. Jika objek superkelas bukan merupakan suatu instans subkelas., error **ClassCastException** akan terjadi. Sebagai contoh, jika suatu objek bukan instans dari **Mahasiswa,** maka objek tersebut tidak bisa di-cast menjadi suatu variabel **Mahasiswa.** Hal ini bisa diselesaikan dengan penggunaan operator instance of

Listing Program

```
public class DemoCasting {
 /** Main method */
 public static void main(String[] args) {
 // Menciptakan dan menginisialisasi dua objek
 Object objek1 = new Lingkaran4(1);
 Object objek2 = new PersegiPanjang1(1, 1);
 // Menampilkan lingkaran dan persegi-panjang
 tampilObjek(objek1);
 tampilObjek(objek2);
 }
 /** Metode untuk menampilkan suatu objek */
 public static void tampilObjek(Object objek){
 if(objek instanceof Lingkaran4) {
 System.out.println("Luas lingkaran adalah " +
```

```
((Lingkaran4)objek).dapatLuas());
System.out.println("Diameter lingkaran adalah " +
((Lingkaran4)objek).dapatDiameter());
}
else if(objek instanceof PersegiPanjang1) {
System.out.println("Luas persegi-panjang adalah " +
((PersegiPanjang1)objek).dapatLuas());
}
}
}
```

Metode **tampilObjek (Object objek)** merupakan contoh pemrograman generik, yang dapat dipanggil degan melewatkan sembarang instans dari **Object.**

Program menggunakan *casting* implisit untuk menugaskan suatu objek **Lingkaran** kepada **objek1** dan suatu objek **PersegiPanjang** kepada **objek2** (baris 5-6), kemudian memanggil metode **tampilObjek()** untuk menampilkan informasi pada kedua objek tersebut (baris 9-10).

Di dalam metode **tampilObjek()** (baris 14-25), casting eksplisit digunakan untuk meng cast objek **Lingkaran** jika objek merupakan suatu instans **lingkaran**, dan metode **dapatLuas()** dan **dapatDiameter()** digunakan untuk menampilkan luas dan diameter suatu lingkaran.

Casting bisa dilakukan hanya jika objek sumber merupakan suatu instans dari kelas target. Program menggunakan operator **instanceof** untuk memastikan bahwa objek sumber merupakan suatu instans dari kelas target sebelum melakukan suatu casting (baris 15).

Casting eksplisit menjadi lingkaran (baris 17,19) dan menjadi **PersegiPanjang** (baris 23) perlu dilakukan karena metode **dapatLuas** dan **dapatDiameter** tidak tersedia di dalam kelas **Object.**

c. Rangkuman

Casting digunakan untuk mengubah variabel-varibel suatu tipe primitif menjadi tipe primitif lain dan digunakan pula untuk untuk mengubah objek dengan suatu tipe kelas menjadi objek dengan tipe kelas lain. Sintaks yang digunakan sama seperti yang digunakan untuk meng-casting tipe primitif, tipe objek target diapit oleh sepasang kurung dan ditempatkan sebelum objek yang akan dicast. Agar casting berhasil dilakukan, Anda perlu memastikan bahwa objek yang akan di-cast merupakan suatu instans subkelas. Jika objek superkelas bukan merupakan suatu instans subkelas, error **ClassCastException** akan terjadi. Casting bisa dilakukan hanya jika objek sumber merupakan suatu instans dari kelas target. Program menggunakan operator **instanceof** untuk memastikan bahwa obyek sumber merupakan suatu instans dari kelas target sebelum melakukan suatu casting.

d. Tugas

Tugas 1

Buat program di java :
Ada suatu kondisi dimana pada tempat fotokopi "MANDIRI Fotocopy" apabila dia pelanggan pada tempat itu maka berapa lembar pun banyaknya dia fotokopi di dapat harga Rp.75,-.
Tapi jika dia bukan pelanggan maka :
a. jika dia fotokopi kurang dari 100 lembar maka dapat harga Rp.150,-
b. jika fotokopi sebanyak 100-200 lembar dapat harga Rp.100,-
c. tapi jika fotokopi lebih dari 200 lembar dia dapat harga Rp.80,-

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

e. Test Formatif

Dalam test ini setiap andaharus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Jelaskan fungsi dari casting objek !
2. Apakah syarat agar casting objek berhasil dilakukan ?

f. Lembar Jawaban Test Formatif (LJ)

LJ- 01 : Jelaskan fungsi dari casting objek !:

....

....

....

....

....

....

LJ- 02 : Apakah syarat agar casting objek berhasil dilakukan ?

....

....

....

....

....

....

....

17. Kegiatan Belajar 19 : Package

a. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 18 siswa diharapkan dapat :

1) Memahami konsep polimorfisme
2) Menyajikan overloading dan overriding dalam class

b. Uraian Materi

1) Package

Package adalah sebuah sarana untuk mengelompokkan atau mengorganisasikan kelas dan *interface* yang sama atau sekelompok menjadi satu unit tunggal dalam *library*. Package mempengaruhi mekanisme hak akses ke kelas didalamnya. Hal terpenting yang diperhatikan pada saat mendeklarasikan package, bahwa *class* tersebut harus disimpan pada suatu *directory* yang sama dengan nama packagenya. Alasan menggunakan package pada java ialah untuk menghindari tabrakan nama kelas yang akan dibuat dengan nama kelas yang sudah ada. Selain itu, salah satu yang menjadi keuntungan menggunakan package adalah untuk mudahnya developer dalam hal mencari dan me-*manage* akses yang diberikan. Mengerti akan konsep dari package akan membantu mengelola dan menggunakan file yang disimpan didalam JAR (Java Archive).

Package juga mempengaruhi mekanisme hak akses ke kelas-kelas di dalamnya.

- ✓ Pengaruh Package terhadap Method main()

  Kelas yang mengandung method main() memilki syarat tidak berada dalam suatu package, dan hirarki posisi foldernya di atas package yang diimport.

- ✓ Membuat Package

  Ada tiga langkah untuk membuat package :

  - ➢ Mendeklarasikan dan memberi nama package.
  - ➢ Membuat struktur dan nama direktori yang sesuai dengan struktur dan nama package.
  - ➢ Mengkompilasi kelas-kelas sesuai dengan packagenya masing-masing.

✓ Mendeklarasikan dan Memberi Nama Package

Deklarasi package harus diletakkan pada bagian paling awal (sebelum deklarasi import) dari *source code* setiap kelas yang dibungkus package tersebut.

Bentuk umum deklarasi package :

```
package namaPackage;
```

Deklarasi tersebut akan memberitahukan kompilator, ke *library* manakah suatu kelas dikompilasi dan dirujuk.

Syarat nama package :

- Diawali huruf kecil,
- Menggambarkan kelas-kelas yang dibungkusnya,
- Harus unik (berbeda dengan nama package standard),
- Merepresentasikan path dari package tersebut
- Harus sama dengan nama direktorinya.

Contoh package standard :

java.lang (berisi kelas-kelas fundamental yang sering digunakan).

java.awt dan *javax.swing* (berisi kelas-kelas untuk membangun aplikasi GUI)

java.io (berisi kelas-kelas untuk proses input output)

✓ Membuat Struktur Direktori

Pada langkah ini, buatlah direktori menggunakan file manager (di windows menggunakan explorer) sesuai struktur package dari langkah sebelumnya. Kemudian tempatkan kelas-kelas tersebut ke direktori yang bersesuaian (mirip seperti menyimpan file-file ke dalam folder).

Package dapat bersarang di package lain, sehingga dapat dibuat hirarki package.

Bentuk umum pernyataan package multilevel :

package namaPackage1[.namaPackage2[.namaPackage3]];

Contoh hirarki package di JDK :

package java.awt.image;

- ✓ Compile dan Run Kelas dari suatu Package

  Selanjutnya masing-masing kelas tersebut dalam package tersebut dikompilasi menjadi byte code (*.class). Artinya package tersebut siap digunakan.

- ✓ Menggunakan Package

  Ada dua cara menggunakan suatu package yaitu :

  - Kelas yang menggunakan berada dalam direktori (package) yang sama dengan kelas-kelas yang digunakan. Maka tidak diperlukan import.
  - Kelas yang menggunakan berada dalam direktori (package) yang berbeda dengan kelas-kelas yang digunakan. Maka pada awal source code di kelas pengguna harus mencantumkan :

    import namaPackage.NamaKelas; atau

    import namaPackage.;*

    Contoh :

    import java.text.DecimalFormat;

    import javax.swing.;*

- ✓ Setting Classpath

  Path hirarki package, didaftarkan sebagai salah satu nilai variabel lingkungan yang bernama Classpath.Classpath diset dengan aturan : berawal dari drive (C:\ atau D:\) sampai dengan satu tingkat sebelum kita mendeklarasikan package.

c. Rangkuman

Package adalah sebuah sarana untuk mengelompokkan atau mengorganisasikan kelas dan *interface* yang sama atau sekelompok menjadi satu unit tunggal dalam *library*. Alasan menggunakan package pada java ialah untuk menghindari tabrakan nama kelas yang akan dibuat dengan nama kelas yang sudah ada.masing-masing kelas tersebut dalam package tersebut dikompilasi menjadi byte code (*.class). Path hirarki package, didaftarkan sebagai salah satu nilai variabel lingkungan yang bernama Classpath. Classpath diset dengan aturan.

d. Tugas

Tugas 1

Buatlah program untuk mencari luas Segitiga dengan mengambil dari package Bangun Datar (misalnya).

❖ Mengamati Listing Program dan Output Program

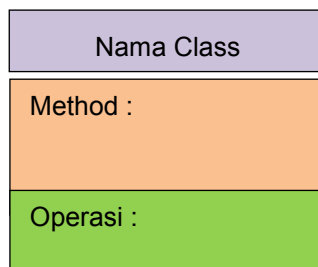
1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method
4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

Tugas 2

Buatlah program untuk menampilkan data siswa di suatu sekolah. Gunakan prinsip *Package* dengan nama Sekolah.

❖ Mengamati Listing Program dan Output Program

1. Menentukan nama Class
2. Menentukan variabel yang digunakan
3. Menentukan nama Method

4. Gambar Class Diagram

Nama Class
Method :
Operasi :

5. Buatlah listing program
6. Compile dan debug program

❖ Bandingkan dan Simpulkan

Bandingkan listing program dan output kelompok Anda dengan Kelompok lain. Berdasarkan hasil perbandingan tersebut hal penting apa yang harus dirumuskan secara bersama?

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	

10.

e. Test Formatif

Dalam test ini setiap andaharus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Bagaimana cara mendeklarasikan dan memberi nama pada package ?
2. Apa yang dimaksud dengan package ?
3. Sebutkan syarat-syarat memberi nama pada package ?
4. Apakah package berpengaruh pada class lainya ? Jika iya berikan alasanya ?

f. Lembar Jawaban Test Formatif (LJ)

LJ- 01 : Bagaimana cara mendeklarasikan dan memberi nama pada package ?

....
....
....
....
....
....
....
....
....

LJ- 02 : Apa yang dimaksud dengan package ?

....
....
....
....
....
....

18. Kegiatan Belajar 20 : Package

a. *Tujuan Pembelajaran*

Setelah mengikuti kegiatan belajar 18 siswa diharapkan dapat :

1) Memahami konsep impor statis dan package terpadu.
2) Menyajikan pengertian tentang impor statis dan package terpadu

b. *Uraian Materi*

Supaya dapat menggunakan *class* yang berada diluar *package* yang sedang dikerjakan, Anda harus mengimport *package* dimana *class* tersebut berada. Pada dasarnya, seluruh program JAVA mengimport *package java.lang*,* sehingga anda dapat menggunakan class seperti String dan Integer dalam program meskipun belum mengimport *package* sama sekali.
Penulisan import *package* dapat dilakukan seperti di bawah ini.

Sintaks Import Packages

```
import <namaPaket>
```

sebagai contoh bila anda ingin menggunakan *class* Color dalam package awt, Anda harus menuliskan import *package* sebagai berikut :

Listing Program

```
import java.awt Color:
import java.awt *;
```

Baris pertama menyatakan untuk mengimport *class* Color secara spesifik pada *package,* sedangkan baris kedua menyatakan mengimport seluruh *class* yang terkandung dalam *package java.aw.*

Cara lain dalam mengimport *package* adalah dengan menuliskan referensi *package* secara eksplisit. Hal ini dilakukan dengan menggunakan nama *package* untuk mendeklarasikan *object* sebuah *class* :

Listing Program

```
Java.awt.Color color,
```

Pada sebuah file.java dibutuhkan referensi file-file mana saja yang menjadi referensi dari class-class, method-method, ataupun segala sesuatu yang digunakan dalam sebuah program java yang ditulis dalam sebuah file.java tersebut, aturan penulisan pada umumnya ditulis di bawah penulisan package, contohnya adalah sebagai berikut :

Listing Program

```
Import java.io.RandomAccesFile;
Import java.net.*;
```

c. Rangkuman

Agar package yang diluar kelas yang dikerjakan dapat digunakan, package tersebut harus di import dahulu. Pada umumnya seluruh program java mengimport *package java.lang.*

d. Test Formatif.

Dalam test ini setiap andaharus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa pengertian dari package ?
2. Apakah package penting di pemrograman java, sebutkan alasanya !

e. Lembar Jawaban Test Formatif (LJ)

LJ- 01 : Apa pengertian dari package ?

....

....

....

LJ- 02 : Apakah package penting di pemrograman java, sebutkan alasanya!

....

....

....

....

Daftar Pustaka

Musnter, Christian. 2006. Java 2 JDK 5 – Grundlagen Herdt – Verlag for bildungsmedien Gagh. Bodenheilm

Wu, C. Thomas. 2001. An Introduction to Object-Oriented Programming with Java. Nort America: McGraw Hill

Deitel. 2002. Java How to program. New jersey. Prentice Hall.

Joyce Avestro. 2003. Introduction Programming 1, Java Education Development Initiatif.

Joyce Avestro. 2003. Introduction Programming 2, Java Education Development Initiatif.

Patric Noughton. 2006. *The Java Handbook*. McGrawHill, Inc.

Sianipar, R.H. 2013. *Teori dan Implementasi Java*. Bandung: Informatika.